HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

DIPLOMARBEIT

Herr

Eric Linke

Analyse und Neukonzeptionierung der Kommunikationsabläufe zwischen der Prüftechnik und dem Labordatenmanagementsystem, sowie Realisierung eines Prototyps

Mittweida, 2013

Fakultät Wirtschaftswissenschaften

DIPLOMARBEIT

Analyse und Neukonzeptionierung der Kommunikationsabläufe zwischen der Prüftechnik und dem Labordatenmanagementsystem, sowie Realisierung eines Prototyps

Autor:
Herr
Eric Linke

Studiengang:
Wirtschaftsingenieurwesen

Seminargruppe:
WI08w4

Erstprüfer:
Prof. Dr. rer. pol. Gunnar Köbernik

Zweitprüfer:
Prof. Dr. Dr. h. c. Hartmut Lindner

Einreichung:
Mittweida, 24.01.2013

Verteidigung/Bewertung:
Mittweida, 2013

Bibliografische Beschreibung:

Linke, Eric:
Analyse und Neukonzeptionierung der Kommunikationsabläufe zwischen der Prüftechnik und dem Labordatenmanagementsystem, sowie Realisierung eines Prototyps
Mittweida, Hochschule Mittweida, Fakultät Wirtschaftswissenschaften, Diplomarbeit, 2013

Referat:

Die vorliegende Arbeit befasst sich mit der Analyse der derzeitigen Kommunikationsabläufe zwischen den Dauerversuchsprüfplätzen und dem Datenbankmanagementsystem der Firma Miele & Cie. KG. Es werden dabei Probleme und prozesskritische Schwachstellen erkannt und anschließend eine Lösung gefunden. Nach der Auswahl eines geeigneten Konzepts und einer Programmiertechnik wird im Anschluss ein erster Prototyp für die veränderte Kommunikation entwickelt. Bei der Lösung werden zukünftig anstehende Änderungen bei der Firma Miele & Cie. KG mit betrachtet.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

DB	Datenbank
DBMS	Datenbankmanagementsystem
DBS	Datenbanksystem
DFVDB	Dauerfeldversuchsdatenbank
ERM	Entity-Relationship-Modell
FMEA	Fehlermöglichkeits- und Einflussanalyse
IDE	integrated development environment; Entwicklungsumgebung
JSP	JavaServer Pages
KVP	Kontinuierlicher Verbesserungsprozess
LabDB	Labordatenbank
PP	Prüfplatz
PPNr	Prüfplatznummer
PrNr	Prüfungsnummer
ProLab	Provo en Laboratorio; Prüfungen im Labor; Anwendungssoftware
QFD	Quality Function Deployment
QM	Qualitätsmanagement
RPZ	Risiko-Prioritäts-Zahl
RZ	Risiko-Zahl
SMV	Steuer- und Messvorschrift
TQM	Total Quality Management
URL	Uniform Resource Locator
VM	virtuelle Maschine

1 Einleitung

1.1 Motivation

Das Unternehmen Miele prüft, getreu dem Firmenmotto „Miele - immer besser“, die Qualität all ihrer Geräte in zahlreichen Tests. Von besonderer Bedeutung ist dabei der Dauerversuch. In diesem werden die Produkte circa 10.000 Stunden im Dauerbetrieb auf Funktion und Belastbarkeit getestet. Nur mit einer solchen Prüfung kann die vom Unternehmen versprochene Produktlebenszeit von mehreren Jahrzehnten gewährleistet werden.[1] Die durchgehend hohe Qualität der Miele-Produkte ist essentiell für den Unternehmenserfolg. Die Kunden verbinden mit der Marke Zuverlässigkeit und Beständigkeit. Es ist also notwendig, die Produktqualität permanent zu beobachten und zu überprüfen.

Die Produkte werden dabei an fest installierte Prüfplätze angeschlossen und diese haben eine Verbindung über eine Anwendungssoftware zu einer Datenbank, in der die Messwerte gespeichert werden. Miele ist allerdings unzufrieden mit der derzeitigen Kommunikation zwischen den Prüfplätzen und der Anwendung, unter anderem aufgrund der hohen Komplexität des derzeitigen Prozesses. Zusätzlich gibt es bereits softwareseitige Veränderungen sowohl innerhalb der Prüfplätze, als auch in der Anwendungssoftware, welche ein Überdenken der derzeitigen Kommunikationsabläufe nötig macht.

1.2 Zielsetzung und Abgrenzung

Ziel der Diplomarbeit ist es, die derzeitige Kommunikation zwischen Prüftechnik und der Datenbank zu analysieren und zu verbessern. Die Ist-Analyse wird Probleme und Schwachstellen innerhalb des aktuell verwendeten Prozesses aufzei-

[1] http://www.miele.de/de/haushalt/unternehmen/4366.htm

gen. Auf Basis derer sollen Lösungskonzepte erstellt werden, welche die Kommunikation optimieren.

Berücksichtigt werden dabei zwei wesentliche Softwareveränderungen im Bereich der Prüftechnik: Einerseits der Wechsel des Betriebssystems der Prüfplätze von RTOS-UH auf QNX und andererseits die Ablösung der Anwendungssoftware DFVDB durch die Miele-Eigenentwicklung ProLab.

Nachdem ein optimierter Kommunikationsprozess gefunden wurde, soll dieser, nach Auswahl der geeignetsten Programmiersprache, in einem Prototyp umgesetzt werden, welcher die Datenübergabe auf einem PC simulieren kann.

Am Standort Gütersloh werden nur Waschmaschinen, Trockner und Waschtrockner geprüft. Alle weiteren Geräte der Produktpalette, wie z.B. Staubsauger und Elektroherde, werden an anderen Standorten getestet und sind nicht Bestandteil dieser Arbeit. Außerdem werden nur die Prüfverfahren des Dauerversuchs analysiert. Andere Prüfungen unterliegen prozessspezifischen Besonderheiten und sind ebenfalls nicht Teil dieser Diplomarbeit.

1.3 Vorgehensweise

Die vorliegende Diplomarbeit beginnt klassisch mit einer Vorstellung des Unternehmens Miele & Cie. KG, sowie der Abteilung Konstruktion und Entwicklung. In Kapitel 2 werden im Anschluss die theoretischen Grundlagen für die weitere Arbeit gelegt. Für die Gliederung dieses Grundlagenkapitels wurde der Kommunikationsprozess in seine drei Hauptbestandteile zerlegt:

Abbildung 1 Die drei Grundelemente des Prozesses

Auf der einen Seite gibt es den **Prüfplatz**, welcher die Aufgabe des Qualitätsmanagements repräsentiert. Daher werden in Kapitel 2 zunächst die Begriffe Qualität, Qualitätsmanagement und Qualitätssicherung definiert und im Rahmen der Qualitätsprüfung die sieben Werkzeuge der Qualität beschrieben.

Auf der anderen Seite steht die **Datenbank**, welche alle Prüfaufträge und -daten für den Prüfplatz bereitstellt und zudem alle entstehenden Werte vom Prüfplatz für die weitere Verarbeitung und Analyse speichert. Daher ist es auch wichtig, diese Thematik näher zu untersuchen. Es werden Begriffe definiert und abgegrenzt, die Notwendigkeit von Datenbanken erklärt, verschiedene Datenbankmodelle verglichen und typische Vertreter vorgestellt.

Im dritten Abschnitt wird die Brücke zu den beiden vorangestellten Säulen Prüfplatz und Datenbank über die **Kommunikation** geschlagen. Damit ein Datenaustausch stattfinden kann, muss eine Technik vorhanden sein, welche diese Übertragung regelt. Da am Ende dieser Diplomarbeit ein Prototyp entstehen soll, ist es wichtig, sich mit der Entwicklung von Programmen auseinanderzusetzen. Dabei werden sowohl auf Vorgangsmodelle als auch auf Programmierparadigmen eingegangen. In der späteren Problembehandlung stehen die drei Programmiersprachen C++, Java und JSP zur Auswahl. Daher werden sie an dieser Stelle zunächst theoretisch beschrieben und miteinander verglichen.

Die in dieser Diplomarbeit bearbeitete Problematik kann als Qualitätsproblem aufgefasst werden. Das heißt, die Kommunikation ist ein Prozess, deren Qualität verbesserungswürdig ist. Unter diesem Gesichtspunkt ist es durchaus sinnvoll, die Problemlösung auch mit Werkzeugen und Methoden des Qualitätsmanagements durchzuführen. Daher werden die nächsten Kapitel nach dem in Kapitel 2.1.2 näher beschriebenen Deming-Kreises bearbeitet. Für diese Diplomarbeit sieht dieser wie folgt aus:

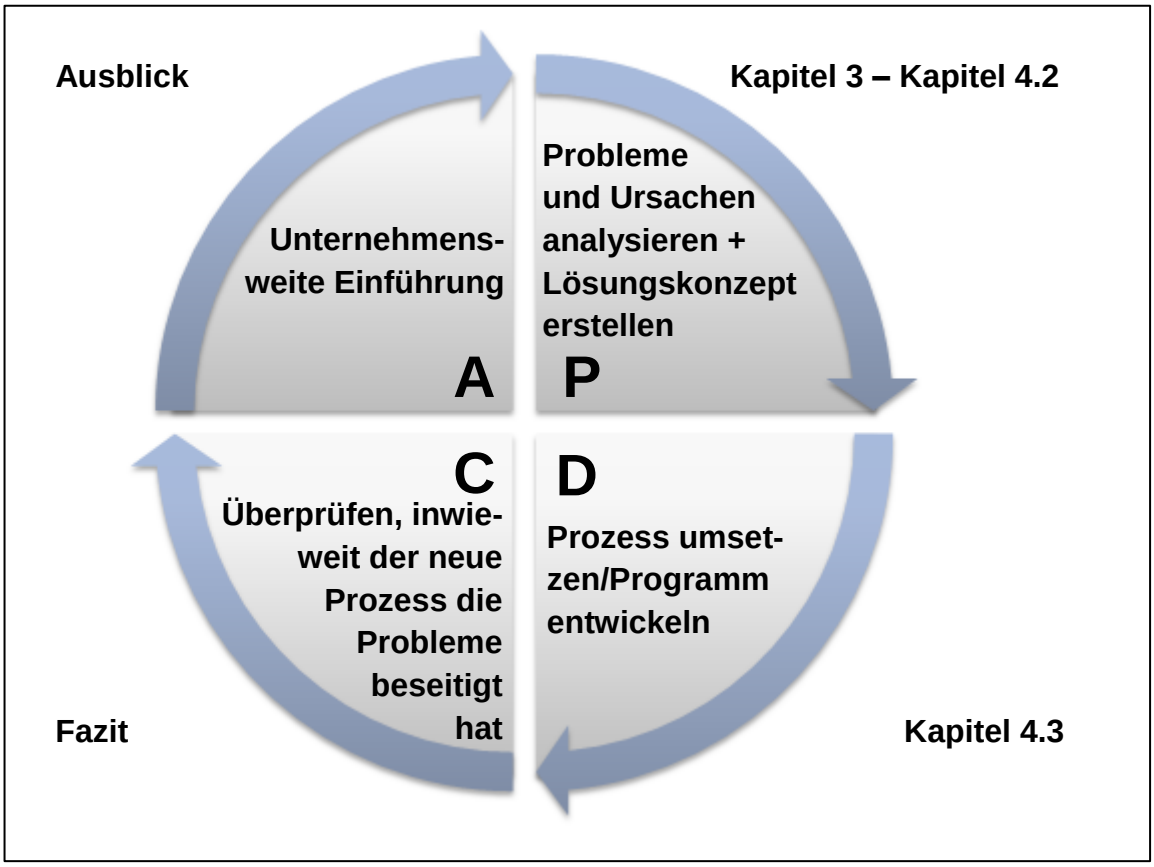


Abbildung 2 Deming-Kreis zur Gliederung

Nach dieser Aufteilung ist der nächste Schritt die Analyse des Ist-Zustandes, welche den aktuellen Stand bei der Firma Miele hinsichtlich des Qualitätsmanagements und der Prüfanordnung beschreibt und die Probleme herausarbeitet. Nach einer Beschreibung der anstehenden Softwareveränderungen folgt in Kapitel 3.5 der Einsatz des in den Grundlagen beschriebenen Qualitätswerkzeuges „Ishikawa-Diagramm“, mit deren Hilfe die Ursachen für diese Probleme analysiert werden.

In Kapitel 4 steht die Problemlösung im Vordergrund, bei der zunächst eine geeignete Programmiersprache ausgewählt und im Anschluss ein auf diese Sprache angepasster optimierter Kommunikationsprozess entwickelt wird. Dabei findet eine Aufteilung des Prozesses in mehrere Module statt und es werden deren Aufgaben und Besonderheiten beschrieben. Damit ist im Deming-Kreis die **P**lan-Phase abgeschlossen.

In Kapitel 4.3 werden die Programmierung und die Besonderheiten, welche während dieser auftauchten, detailliert beschrieben. Diesen Abschnitt symbolisiert die **D**o-Phase.

In der **C**heck-Phase soll überprüft werden, inwieweit der neue Prozess die aufgezeigten Probleme behebt. Dafür wird im Fazit das in Kapitel 3.5 erstellte Ishikawa-Diagramm erneut verwendet und hinsichtlich des neuen Prozesses überprüft. Auf diese Weise ist auf einen Blick das Resultat der Arbeit erkenntlich.

Am Ende wird in einem kurzen Ausblick beschrieben, wie der Prozess in Zukunft Anwendung findet. Da diese Diplomarbeit mit der Erstellung eines Prototyps abgeschlossen ist, folgen einige Hinweise für die unternehmensweite Einführung. Im Deming-Kreis wird dies mit dem **A** für Act dargestellt.

1.4 Unternehmensprofil

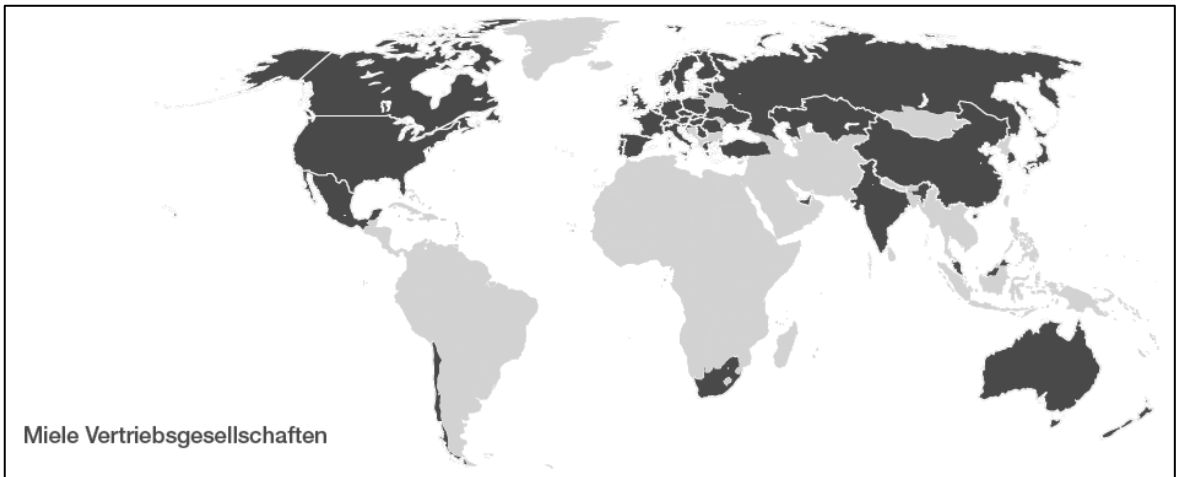
Die Firma Miele & Cie. KG wurde 1899 von Carl Miele und Reinhard Zinkann gegründet. In Herzebrock, nahe Gütersloh, begann die Firma Miele zunächst mit der Produktion von Milchzentrifugen. Recht schnell wurde das Sortiment auch um eine Buttermaschine und die erste Oberpendel-Waschmaschine erweitert.

Heute gilt das in Gütersloh ansässige Unternehmen Miele als einzige weltweit verbreitete Premium-Marke für Hausgeräte, die auf fünf Kontinenten vertrieben wird.[2] Dabei wird besonderen Wert auf Nachhaltigkeit, Innovationen und die Qualität der Produkte gelegt. Die Produktpalette reicht dabei heute von Wasch- und Trockenautomaten über Staubsauger, Geschirrspüler, Kochgeräte und Kaffeevollautomaten. Auch auf gewerblicher Ebene stellt das Unternehmen Produkte im Bereich der Wäschereitechnik, Geschirrspüler-, Labor- und Medizintechnik zur Verfügung.

[2] Vgl. http://miele.de/de/haushalt/unternehmen/4270.htm

Produziert werden die Produkte weltweit in 11 Werken. Davon sind 8 Standorte in Deutschland ansässig, sowie ein Werk jeweils in Österreich, Tschechien und China. Bei der Herstellung wird dabei ein hoher Wert auf die Eigenproduktion gelegt. So wird beispielsweise die Elektronik nicht kostengünstig eingekauft, sondern selbst im firmeneigenen Elektronikwerk in Gütersloh produziert. So kann Miele den hohen Qualitätserwartungen gerecht werden.

Vertrieben werden die Produkte in 47 Vertriebsgesellschaften in ebenso vielen Ländern. Der Jahresumsatz wuchs laut Geschäftsbericht im Geschäftsjahr 2011/2012 auf 3,04 Milliarden Euro. Dies ist der höchste, in der Firmengeschichte erreichte Wert. In diesem Zeitraum wurden zudem Investitionen in Höhe von 186 Millionen Euro getätigt. Weltweit arbeiten derzeit 16.700 Mitarbeiter für das Unternehmen Miele.[3]



	2007/08	2008/09	2009/10	2010/11	2011/12
Umsatz* weltweit	2,81	2,77	2,83	2,95	3,04
davon in Deutschland	0,74	0,80	0,84	0,87	0,91
Veränderung zum Vorjahr	+2,5 %	−1,3 %	+2,2 %	+4,0 %	+3,1 %
Anzahl Beschäftigte	16.163	16.555	16.561	16.624	16.716

* in Mrd. Euro

Abbildung 3 Umsatz- und Mitarbeiterentwicklung der Miele Gruppe[4]

[3] http://www.miele-presse.de/media/presse/media/Miele_GB_2011-12_DE_web.pdf

[4] http://www.miele-presse.de/media/presse/media/Miele_GB_2011-12_DE_web.pdf S.5

1.5 Abteilung Konstruktion und Entwicklung

Für die Dauerversuche der Wasch- und Trocknerautomaten ist in der Firma Miele die Abteilung KEW (Konstruktion und Entwicklung) zuständig. Der Aufgabenbereich dieser Abteilung umfasst dabei folgendes Spektrum:

- Entwicklung Elektrotechnik,
- Entwicklung Gehäuse,
- Entwicklung/Labore Waschen und Entwicklung/Labore Trocknen,
- Musterbau,
- Produktentwicklung Waschen/Trocknen,
- Vorentwicklung/Technologie.

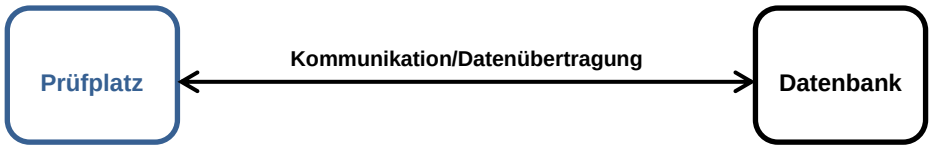
Die Messgeräte und Software für die KEW werden von der Gruppe „Labordienstleistungen“ bereitgestellt und die kompletten Mess- bzw. Prüfplätze eingerichtet. Außerdem ist diese Gruppe auch für die Speicherung der Messdaten auf der Datenbank, sowie die Zustandsvisualisierung zuständig.

Die Abteilung Konstruktion und Entwicklung ist direkt der Werkleitung unterstellt.[5]

[5] Miele-Intranet: http://mielewss.de.miele.net/wss/puma/Anweisung/01304-01%20Funktionsbeschreibung%20Konstruktion-Entwicklung%20(GTG-KEW)/01304-01.pdf

2 Einführung in das Qualitätsmanagement und die Informationstechnologie

2.1 Qualitätsmanagement



2.1.1 Qualität – Begriffsdefinition und Notwendigkeit

> „Zukünftig werden hochentwickelte Technologien weniger entscheidend sein als vielmehr die Einstellung eines Unternehmens, Kundenerwartungen erfüllen zu wollen.“[6]

Einige der bekanntesten Denker und Philosophen wie Lao-Tse, Sokrates, Platon, Aristoteles, Leibniz und Kant haben versucht, den Begriff „Qualität“ umfassend zu beschreiben.[7] Allerdings werden diese Definitionen der heutigen Anwendung im Produktionsbereich nicht gerecht.

Die Wortherkunft des Begriffs „Qualität“ lässt sich auf das lateinische Wort „qualitas“ zurückführen, was so viel bedeutet wie „Beschaffenheit“ oder „wie beschaffen“.[8]

Im Einklang zu dieser Übersetzung definiert Geiger Qualität wie folgt:

Qualität = Relation zwischen realisierter Beschaffenheit und geforderter Beschaffenheit[9]

---

[6] Timischl, Wolfgang; Qualitätssicherung: Statistische Methoden

[7] Zollondz, Hans-Dieter; Grundlagen Qualitätsmanagement; S.8 ff

[8] http://www.duden.de/rechtschreibung/Qualitaet

Mit anderen Worten ist Qualität das Verhältnis aus Ist- und Soll-Beschaffenheit und nicht, wie oft behauptet, eine besonders gute Beschaffenheit. Eine hohe Qualität bedeutet demnach, dass die realisierten Eigenschaften/Ergebnisse den Erwartungen/Anforderungen besonders gut entsprechen.

Geiger veranschaulicht seine Qualitätsdefinition in folgender Grafik als Waagschalen:

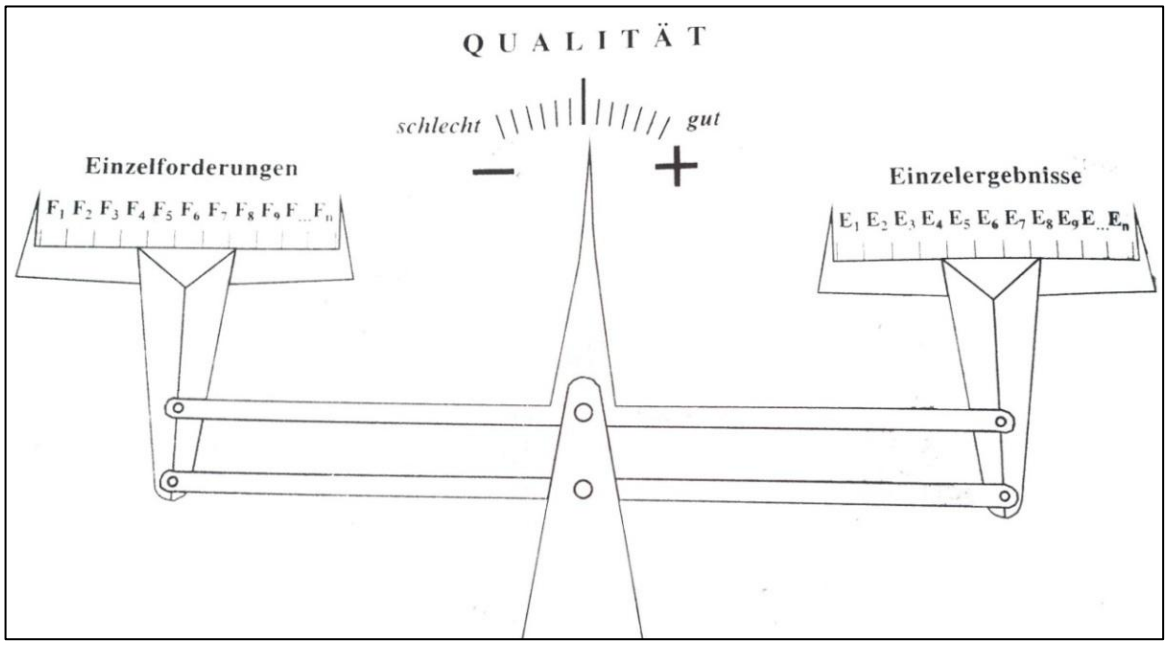


Abbildung 4 Qualität nach Geiger[10]

Hierbei wird die Gesamtqualität in mehrere Qualitätsmerkmale unterteilt. Je mehr Einzelforderungen den Einzelergebnissen gleichen oder übertreffen, umso besser ist die gesamte Qualität.

Laut DIN EN ISO 8402 wird die Qualität wie folgt beschrieben:

> „Die Gesamtheit von Merkmalen einer Einheit bezüglich ihrer Eignung, festgelegte und vorausgesetzte Erfordernisse zu erfüllen“.[11]

---

[9] Geiger, Walter; Kotte, Willi; Handbuch Qualität: Grundlagen und Elemente des Qualitätsmanagements: Systeme – Perspektiven; S. 68

[10] Zollondz, Hans-Dieter; Grundlagen Qualitätsmanagement; S.150

[11] http://quality.kenline.de/seiten_d/qualitaet_definition.htm

Ist ein Produkt von schlechter Qualität, so erfüllt es die Erwartungen des Käufers bzw. Nutzers nicht im genügenden Maße.

Seit Mitte des 20. Jahrhunderts wird die Qualität zunehmend wissenschaftlich betrachtet. Die Qualitätswissenschaft ist nach Gerd F. Kamiske ein interdisziplinäres Fachgebiet, welches sowohl Ingenieurs-, Wirtschafts-, Umwelt- und Sozialwissenschaften beinhaltet. Es vereint demnach sowohl technische, ökonomische, ökologische und humane Aspekte in sich.

Die Abbildung 5 zeigt die Qualitätswissenschaft als Querschnittsdisziplin[12]:

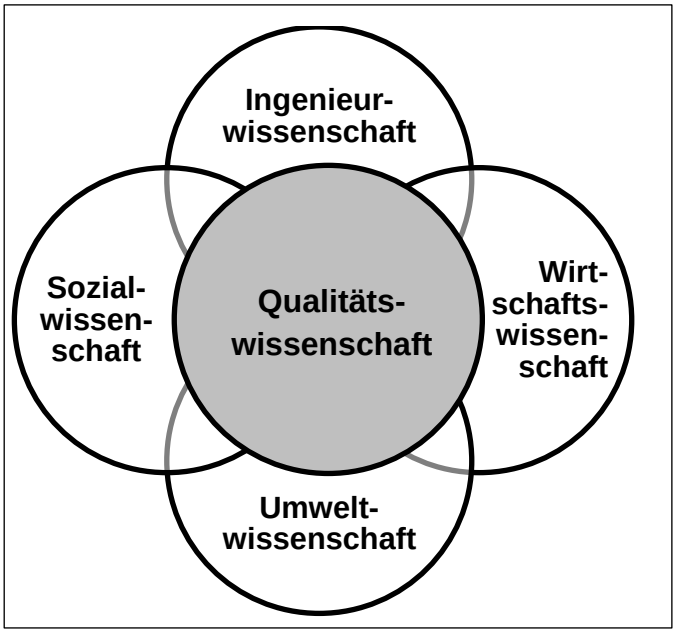


Abbildung 5 Qualitätswissenschaft

Aus diesem interdisziplinären Ansatz heraus lässt sich in der praktischen Umsetzung ein Managementkonzept erarbeiten, das sogenannte Qualitätsmanagement.

2.1.2 Qualitätsmanagement/Total Quality Management

Zur Gewährleistung einer optimalen Qualität darf nicht nur die Fehlererkennung im Vordergrund stehen, sondern auch die Fehlervermeidung. Denn Qualität ist eine

[12] Zollondz, Hans-Dieter; Grundlagen Qualitätsmanagement; S. 22

Eigenschaft, die von vornherein geplant werden muss. Fehler die nicht gemacht werden, müssen später auch nicht behoben werden. Dieser Umstand wird besonders bewusst, wenn man sich den Verlauf der Kosten pro Fehler während des Produktlebenszyklusses anschaut:

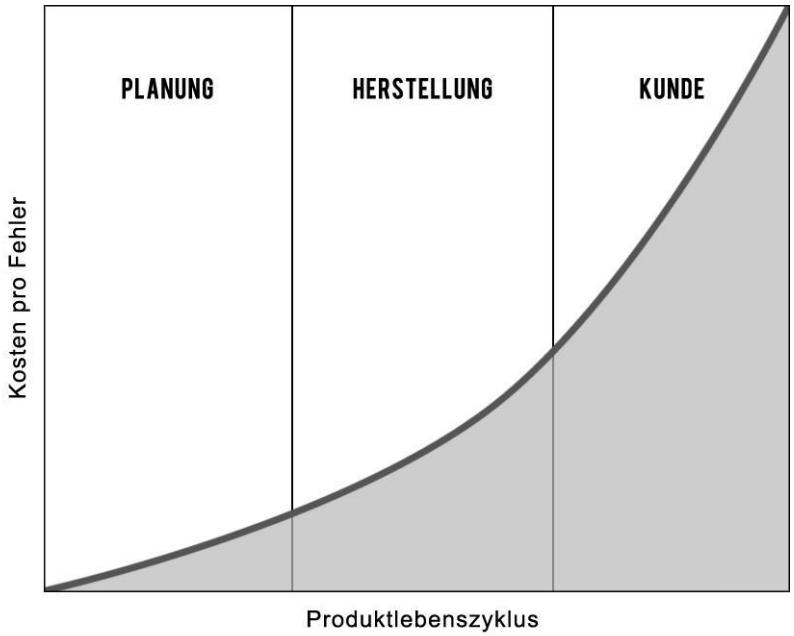


Abbildung 6 Fehlerkostenentwicklung[13]

Die Kosten für die Beseitigung von Fehlern steigen mit fortlaufendem Entwicklungsverlauf um den Faktor 10 (10er-Regel der Fehlerkosten)[14]. Daher sollte ein Unternehmen stets nach dem Leitspruch „Fehlervermeidung ist besser als Fehlerbeseitigung“ handeln[15].

All Maßnahmen zur Verbesserung und Einhaltung der Qualität werden **Qualitätsmanagement** genannt. Dieses Vorgehen wurde mehr und mehr in die Unternehmensphilosophien einbezogen und ist zu einem unternehmensweiten Denken und Handeln geworden, indem alle Mitarbeiter und Bereiche involviert sind. Auch wird sich bei den Maßnahmen immer stärker am Kunden orientiert. Die vollkommene

[13] www.anoxa.de

[14] http://www.anoxa.de/blog2/?p=751

[15] Tietjen, Thorsten; Müller, Dieter H.; FMEA-Praxi; S. 3

Einführung des Qualitätsmanagements in die Unternehmensphilosophie wird als **Total Quality Management (TQM)** bezeichnet.

Thomas Hummel definiert Total Quality Management folgendermaßen:

> „TQM ist eine auf der Mitwirkung aller ihrer Mitglieder basierende Managementmethode einer Organisation, die Qualität in den Mittelpunkt stellt und durch Zufriedenstellung der Kunden auf langfristigen Geschäftserfolg sowie auf Nutzen für die Mitglieder der Organisation und für die Gesellschaft zielt.“[16]

Der Begriff „Total“ gibt dabei an, dass möglichst alle Teilnehmer am Prozess mit einbezogen werden sollten. Dies sind nicht nur die Mitarbeiter, sondern auch die Kunden und Lieferanten. Erst wenn die Qualität hinsichtlich aller Zielgruppen überprüft wird, kann sie als optimiert angesehen werden.

Das Wort „Management“ deutet darauf hin, dass Qualität nicht mehr nur als Produkteigenschaft gesehen werden darf. Es ist vielmehr eine Führungsaufgabe. Erst durch konstantes Planen und Handeln kann ein optimaler Prozess mit maximaler Qualität entstehen.

Innerhalb des TQM-Konzeptes findet meist der **Kontinuierliche Verbesserungsprozess (KVP)** Anwendung. KVP ist eine Denkweise zur stetigen Verbesserung in kleinen Schritten. Dabei wird sich sowohl auf die Prozess-, Produkt- und Servicequalität konzentriert.

Als Werkzeug für das KVP wird oft der Demingkreis bzw. PDCA-Zyklus verwendet. Deming geht davon aus, dass jede Aktivität als Prozess angesehen und somit verbessert werden kann. Die Prozesse werden dann innerhalb von vier Phasen schrittweise optimiert.

[16] Hummel, Thomas; Total Quality Management, S. 5

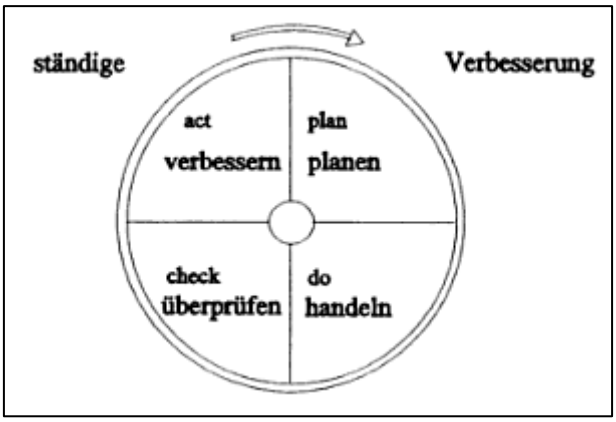


Abbildung 7 PDCA-Zyklus[17]

Die einzelnen Phasen sind[18]:

Plan: Probleme müssen erkannt und deren Ursachen identifiziert werden. Außerdem werden Ziele festgelegt.

Do: Maßnahmen zur Lösung werden erarbeitet und testweise umgesetzt.

Check: Es findet ein Soll-Ist-Vergleich statt und es wird überprüft, ob das Problem gelöst wurde.

Act: Anpassung und Veränderung des Prozesses auf breiter Front bzw. unternehmensweite Einführung der Verbesserung.

2.1.3 Qualitätssicherung

Die Qualitätssicherung ist laut Prof. Dr. Voigt als „Bestandteil des Qualitätsmanagements“ definiert, welches „alle organisatorischen und technischen Maßnahmen umfasst, die vorbereitend, begleitend und prüfend der Schaffung und Erhaltung einer definierten Qualität eines Produkts oder einer Dienstleistung dienen.“ [19]

[17] Timischl, Wolfgang; Qualitätssicherung; S. 3

[18] Weigert, Johann; Der Weg zum leistungsstarken Qualitätsmanagement; S. 69

[19] Prof. Dr. Voigt, Kai-Ingo; http://wirtschaftslexikon.gabler.de/Definition/qualitaetssicherung.html

Den Zusammenhang zwischen Qualitätssicherung und Total Quality Management hat Wolfgang Timischl in seinem Buch „Qualitätssicherung“ sehr anschaulich mit folgender Grafik hergestellt:

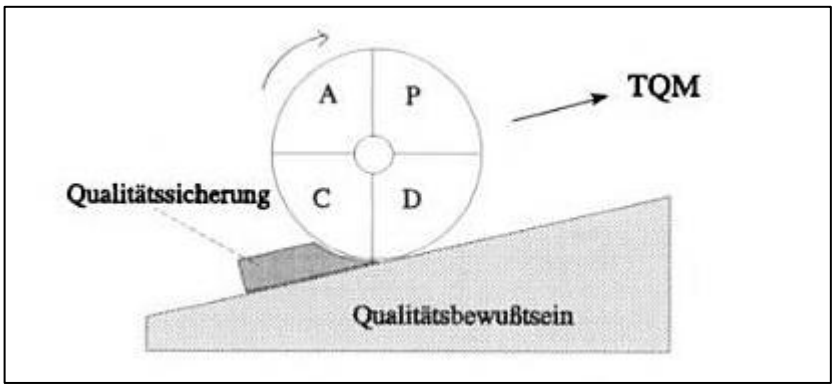


Abbildung 8 Zusammenhang Qualitätssicherung - TQM[20]

Das Deming-Rad wird dabei von allen Mitarbeitern eines Unternehmens eine schiefe Ebene hinauf gerollt, welches das Qualitätsbewusstsein symbolisiert. Nur durch permanentes Drehen des PDCA-Rads geht der Prozess voran und der Berg wird erklommen und somit die Qualität gesteigert. Die Qualitätssicherung dient als Stopper, um ein zurückrollen zu vermeiden.

2.1.4 Qualitätsprüfung

Eine Qualitätssteigerung ist nur dann möglich, wenn ein Produkt oder ein Prozess ein Steigerungspotential aufweist (in der Abbildung 8 symbolisiert durch die schiefe Ebene). Das heißt, es muss ein Problem vorhanden sein, welches gelöst werden kann bzw. es muss eine Verbesserung im Prozess möglich sein. Bevor man sich um die Steigerung der Qualität kümmern kann ist es also erst einmal vonnöten, Potentiale zu erkennen.

Bei Produkten geschieht dies meist durch eine Vielzahl von Prüfungen seitens des Herstellers. Dabei werden Alltags- wie auch Extremsituationen nachgestellt und

[20] Timischl, Wolfgang; Qualitätssicherung; S. 4

Produkteigenschaften unter definierten Bedingungen überprüft. Dabei kann sowohl einmalig der Qualitätsgrad ermittelt, als auch regelmäßig die Einhaltung von Qualitätsstandards überprüft werden. Hierfür müssen sich jeweils auf vorgegebene Grenzen geeinigt werden, in denen sich die Prüflinge bewegen dürfen. Fallen sie unter diese Grenze, gelten sie als „Nicht in Ordnung“ (NiO) bzw. als „in Ordnung“ (iO), wenn sie über der Qualitätsgrenze liegen.

Bei der technischen Qualitätssicherung kann man zwischen der statischen und der dynamischen Qualitätskontrolle unterscheiden. Die statische Kontrolle hat dabei festgelegte Qualitätsvorgaben, welche das Produkt erfüllen muss und welche von externen Stellen geprüft werden. Bei der dynamischen Qualitätskontrolle existieren keine Vorgaben und die Organisation legt die Ziele selbst fest. Die eigenverantwortliche Entwicklung steht hier im Vordergrund.

Zunächst werden oft verwechselte Begriffe geklärt, da diese im allgemeinen Sprachgebrauch fälschlicherweise als Synonyme verwendet werden.

- **Prüfen** bedeutet, dass ein Prüfgegenstand auf die Einhaltung von vereinbarten, erwarteten oder vorgeschriebenen Bedingungen untersucht wird. Dabei wird zwischen subjektiver Prüfung (Prüfen ohne Messgeräte, sondern mithilfe der Sinne z.B. Sichtprüfung) und objektiver Prüfung (Prüfen mit Messgeräten/Prüfmitteln) unterschieden. Die objektive Prüfung kann weiterhin in Messen und Lehren unterteilt werden.

- **Messen** bezeichnet einen experimentellen Vorgang mit dem Ziel, den Wert einer physikalische Größe (Messgröße) zu bestimmten. Dieser Wert wird in einem Vielfachen einer Maßeinheit. Mehrere Messwerte ergeben ein Messergebnis. Ein solches Messergebnis beinhaltet zudem die Messbedingungen und die Messunsicherheit. Das Messen kann man nach zwei Kategorien unterscheiden:
    - direkte und indirekte Messverfahren
    - analoge und digitale Messungen[21]

[21] Pfeifer, Tilo; Qualitätsmanagement – Strategien, Methoden, Techniken; S.493

- **Lehren** ist ein Verfahren, welches prüft, ob ein bestimmtes Merkmal innerhalb bestimmter Grenzen liegt. Dabei wird allerdings nicht angegeben, wie weit der Wert von der Vorgabe abweicht.

Nachdem eine Prüfung abgeschlossen und ausgewertet wurde, werden eventuelle Probleme und Verbesserungspotentiale deutlich. Die Behebung dieser Probleme wird zu einer Steigerung der Qualität führen. Da ein Fehler umso teurer für den Hersteller ist, je später er erkannt wird, sollte ein Produkt nicht nur während der Serienproduktion einer permanenten Qualitätsprüfung unterzogen werden, sondern bereits während der kompletten Entwicklungsphase.

2.1.5 Die sieben Werkzeuge der Qualität

Neben der Qualitätsprüfung gibt es im Qualitätsmanagement auch die sogenannten sieben Werkzeuge der Qualität, welche von Ishikawa Karou 1943 zusammengestellt wurden. Diese umfassen sieben Methoden, welche Probleme, sowie deren Ursachen und Lösungsmöglichkeiten in Prozessen und Produkten aufzeigen. Sie stellen eine Hilfe zur Strukturierung und Visualisierung komplexer Fragestellungen dar und unterstützen damit alle Phasen des PDCA-Kreises.[22]

Ishikawa-Diagramm

Das Ishikawa-Diagramm (auch als Ursache-Wirkungs- oder Fischgrätendiagramm bekannt) ist eine grafische Methode, um Probleme hinsichtlich ihrer Haupt- und Nebenursachen zu analysieren. Als Hauptursachen werden dabei die sogenannten **5M** verwendet. Diese sind:

- **M**ensch
- **M**aschine
- **M**ethode

[22] Pfeifer, Tilo; Qualitätsmanagement – Strategien, Methoden, Techniken; S.40

- **M**aterial
- **M**itwelt (Umwelt)

Oft werden diese fünf Hauptgruppen um die weiteren zwei Bereiche

- **M**anagement und
- **M**essbarkeit

erweitert.

Jeder dieser Hauptursachen können nun mehrere Nebenursachen zugeordnet werden. Auch diese können weitere tiefergehende Ursachen haben, ähnlich einer Mindmap[23]. So wird nach und nach den Ursachen des Problems auf den Grund gegangen und von allen Seiten beleuchtet.

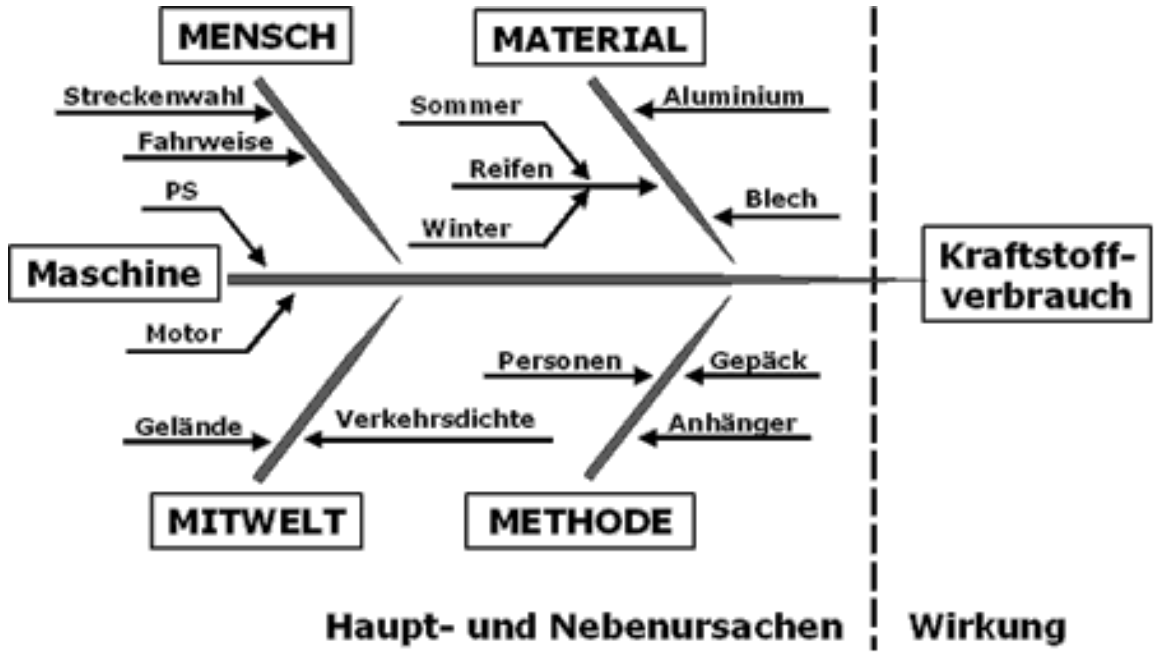


Abbildung 9 Ishikawa-Diagramm[24]

[23] REFA: Ausgewählte Methoden zur prozessorientierten Arbeitsorganisation; S. 711

[24] http://www.wahlen-web.info/beruf_sites/ishikawa.gif

Histogramm

Das Histogramm dient zur Darstellung der Verteilung der Messdaten und ermöglicht damit eine Interpretation von Streuungsursachen. Die Messwerte werden dafür in Klassen eingeteilt, welche eine konstante oder variable Breite haben und in der Abszisse des Diagramms eingebunden werden. Auf der Ordinate wird dann die Anzahl der Messwerte pro Klasse abgebildet. Aus der Verteilungskurve lassen sich anschließend Mittelwert und Art der Streuung ableiten[25].

Die Vorteile eines Histogramms sind sowohl die übersichtliche Darstellung großer Datenmengen, als auch die daraus resultierenden möglichen Rückschlüsse[26]. So kann die Verteilung der einzelnen Klassen leicht abgelesen werden und Probleme werden schnell offensichtlich.

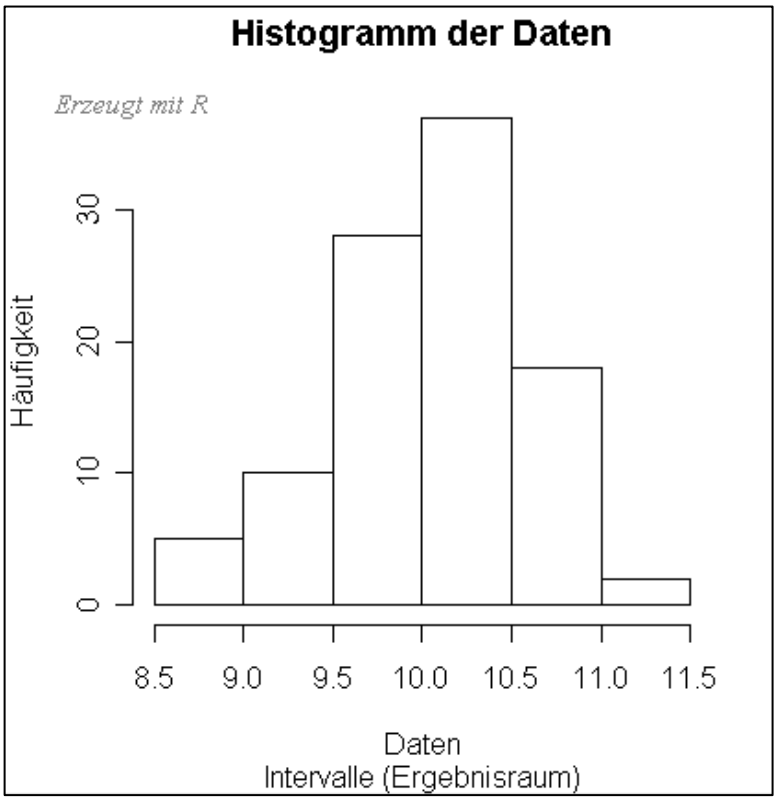


Abbildung 10 Histogramm[27]

[25] Pfeifer, Tilo; Qualitätsmanagement – Strategien, Methoden, Techniken; S.40

[26] Linß, Gerhard; Qualitätsmanagement für Ingenieure; S. 114

[27] http://www.faes.de/Basis/Basis-Lexikon/Basis-Lexikon-Histogramm/Histogramm_1.gif

Korrelations-Diagramm

Das Korrelations-Diagramm stellt grafisch die Beziehung zwischen zwei Merkmalen dar. Dabei werden die Wertepaare im Diagramm als Punkt dargestellt. Aus dem entstehenden Muster lassen sich dann Rückschlüsse auf einen statistischen Zusammenhang zwischen den beiden Merkmalen ziehen.

Es ist auch eine sinnvolle Ergänzung zum Ishikawa-Diagramm, da man so grafisch den Zusammenhang zwischen Ursache und Wirkung darstellen kann. Es kann allerdings nur ein statistischer, nicht jedoch ein kausaler Zusammenhang ermittelt werden[28].

Je nach entstehendem Muster können sechs verschiedene Korrelationsformen unterschieden werden:

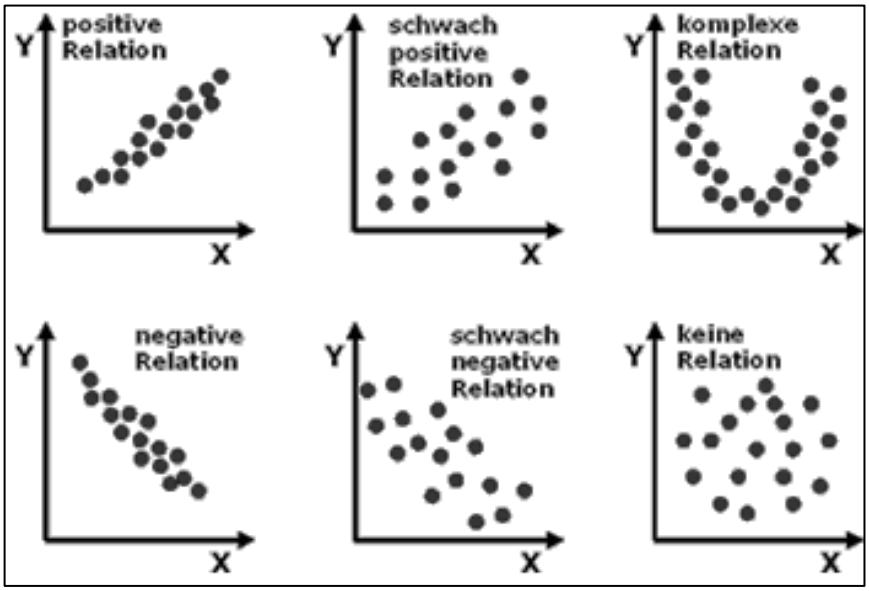


Abbildung 11 Korrelations-Diagramm[29]

[28] Colsman, Hubertus; Theden, Philipp; Qualitätstechniken: Werkzeuge zur Problemlösung und ständigen Verbesserung; S. 34

[29] http://www.wahlen-web.info/beruf_sites/korrelationsformen.gif

Brainstorming

Das Brainstorming ist eine der bekanntesten Kreativitätstechniken bzw. Ideenfindungsmethode. Sie kann sehr gut zur Lösungsfindung von Problemen genutzt werden. Dafür wird das Problem von einem Moderator klar definiert und in der ersten Kreativphase alle Gedanken und Ideen der Gruppe, ohne jegliche Wertung gesammelt. Dabei kommt es hauptsächlich auf eine möglichst große Anzahl von Ideen und Vorschlägen an und weniger auf deren Qualität. In der zweiten Phase werden die gesammelten Ideen sortiert und bewertet. Das Ziel ist, sich von gewohnten Denkschemata zu lösen und schnell zu Ergebnissen zu kommen.

Die spontanen Lösungsvorschläge müssen nicht immer einer optimalen Lösung entsprechen. Daher ist diese Methode hauptsächlich für einfache und überschaubare Probleme geeignet. Zu beachten ist außerdem, dass der Moderator strikt gegen destruktive Kommentare während der Kreativphase vorgehen muss, da diese die Kreativität der Gruppe stark hemmen.[30]

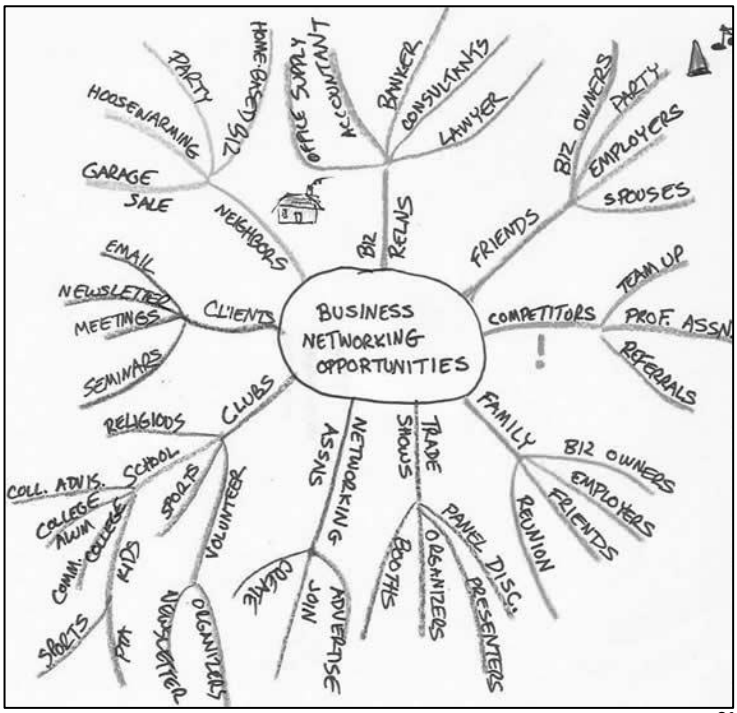


Abbildung 12 Brainstorming am Beispiel einer Mindmap[31]

[30] Brauer, Jörg-Peter; Kamiske, Gerd F.; Qualitätsmanagement von A - Z: Erläuterungen moderner Begriffe des Qualitätsmanagements; S. 250

[31] http://wiseupmarketing.files.wordpress.com/2011/10/brainstormbyyourself.jpg

Regelkarte

Die (Qualitäts-) Regelkarte ist ein statistisches Werkzeug zur Überwachung von Prozessen. Dabei werden während des zu prüfenden Prozesses fortlaufend Stichproben gezogen und in der Regelkarte erfasst. Es werden Toleranzgrenzen festgelegt und die Eintragungen zeigen, ob die Stichproben diese Grenzen einhalten. Solang alle Messergebnisse innerhalb der festgesetzten Toleranzgrenze/Eingriffsgrenze liegen, gilt der Prozess als beherrscht[32], andernfalls muss eventuell in den Prozess eingegriffen werden.

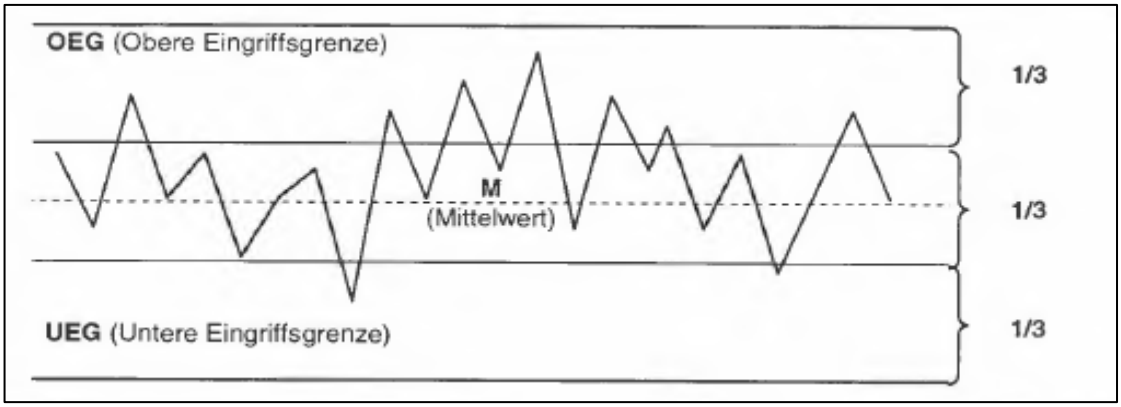


Abbildung 13 Regelkarte[33]

Pareto-Analyse

Bei einer Prozessanalyse sieht man sich oft einer Vielzahl von Problemen und Fehlerursachen gegenüber gestellt. Dabei sind allerdings nicht alle gleichermaßen schwerwiegend und treten nicht gleich oft auf. Das größte, wichtigste oder kostenintensivste Problem sollte zuerst gelöst werden. Zur Entscheidungshilfe, welches dies ist, dient die Pareto-Analyse. In dieser werden zunächst alle Problem- oder Fehlerarten erfasst und deren Häufigkeit des Auftretens zugeordnet. So entsteht eine grafische Darstellung der Fehlerarten, sortiert nach deren Auftrittshäufigkeit. Nun können schwerwiegende Probleme schnell erfasst und behoben werden. Die

[32] REFA: Ausgewählte Methoden zur prozessorientierten Arbeitsorganisation; S. 708

[33] REFA: Ausgewählte Methoden zur prozessorientierten Arbeitsorganisation; S. 708

Pareto-Analyse ist eine Entscheidungshilfe und keine Hilfestellung bei der Problemlösung[34].

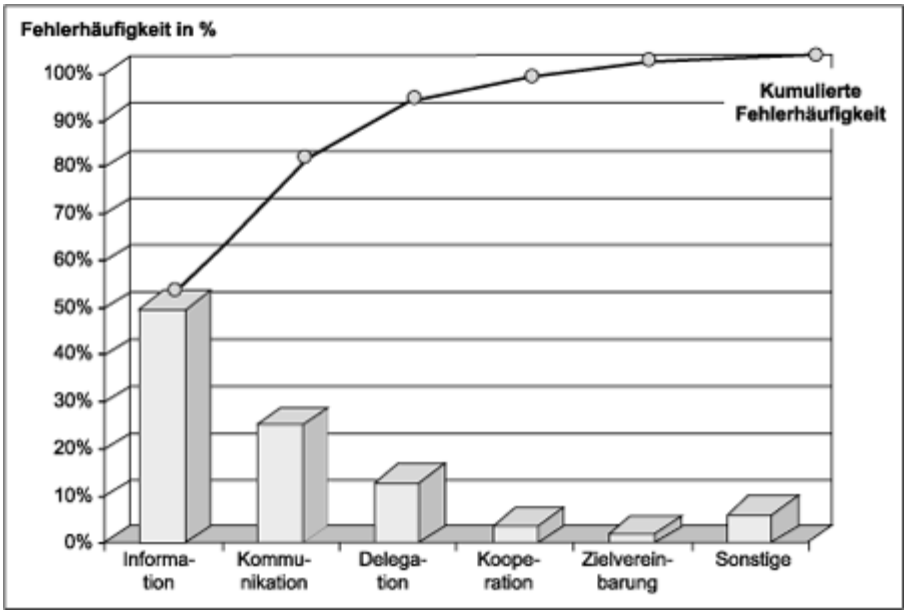


Abbildung 14 Pareto-Analyse[35]

Strichliste

Mit Hilfe der Strichlisten ist es auf einfache Art und Weise möglich, Häufigkeiten von Fehlern in bestimmten Intervallen darzustellen. Dazu werden meist vorbereitete Formulare verwendet auf denen alle erwarteten oder aufzunehmenden Fehler aufgelistet sind. Bei Auftreten eines solchen Fehlers wird eine Markierung gesetzt.

Fehler	Häufigkeit	Summe
A	~~IIII IIII~~	10
B	III	3
C	~~IIII IIII~~ II	12
D	~~IIII~~ III	8
E	III	3
Summe		36

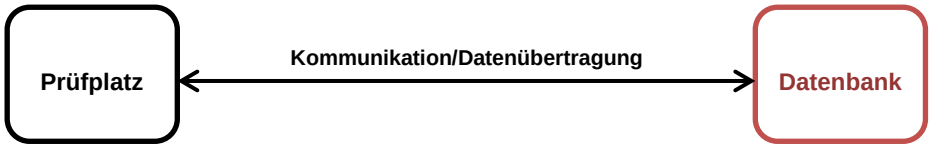
Abbildung 15 Strichliste[36]

[34] Pfeifer, Tilo; Qualitätsmanagement – Strategien, Methoden, Techniken; S.41

[35] http://schiering.org/arhilfen/qualit/qm/1103_03.gif

[36] Anlehnung an Pfeifer, Tilo; Qualitätsmanagement – Strategien, Methoden, Techniken; S.41

2.2 Datenbanksysteme



In einer Datenbank werden bei der computergestützten Prüfung alle Daten gespeichert. Dies sind sowohl die Prüfparameter, welche für die Prüfung notwendig sind, als auch die Prüfdaten, die während einer Prüfung entstehen und im Anschluss ausgewertet werden.

Bei der Datenspeicherung unterscheidet man zwei Varianten: Einerseits die Datenspeicherung mittels **Dateisystem**, also die Möglichkeit, dass Programme ihre benötigten Daten in eigene Dateiformate jeweils unabhängig von anderen Programmen ablegen können. Jedes Programm speichert seine Daten in eigene Dateien deren Aufbau nur dieser Anwendung bekannt ist. Die Daten sind dann den Programmen fest zugeordnet. Dem gegenüber steht das Konzept der **Datenbanken**, bei denen das Abspeichern von Daten an einem zentralen und externen Ort stattfindet, auf dem mehrere Programme zugreifen können.[37]

Die direkte Verknüpfung von Daten und Anwendung wirkt logisch und unproblematisch, da so eine anforderungsoptimierte Datenstruktur für das einzelne Programm gewährleistet wird. Jedoch birgt es bei genauerem Hinsehen viele Nachteile, welche gerade bei der Nutzung mehrerer Programme deutlich werden. Diese Nachteile sind nach Markus Schneider folgende:[38]

Ein großes Problem stellt die sogenannte **Redundanz** dar. Wenn jedes Programm seine Daten für sich selbst in seinem eigenen Format abspeichert, kommt es unvermeidlich dazu, dass Daten mehrfach abgespeichert werden. Daraus resultiert nicht nur eine unnötig höhere Datenmenge, sondern auch ein drastischer Anstieg des Verwaltungsaufwandes, denn all diese mehrfach vorhandenen Daten müssen auch gleichermaßen gepflegt werden.

---

[37] Sauer, Hermann; Relationale Datenbanken: Theorie und Praxis; S. 24

[38] Schneider, Markus; Implementierungskonzepte für Datenbanksysteme; S.2

Dies führt zum nächsten Problem: der Daten-**Inkonsistenz**. Dadurch, dass alle Programme ihre Daten separat abspeichern, ist eine Übereinstimmung der Dateninhalte nicht mehr gewährleistet. Wenn zum Beispiel ein Datumswert inkorrekt ist, muss dies in jeder einzelnen Datei korrigiert werden. Außerdem muss sichergestellt werden, dass Anwendungen nicht zur selben Zeit auf unterschiedliche Versionen des geänderten Datumswertes zugreifen können.

Des Weiteren gibt es in der herkömmlichen Datenspeicherung eine hohe **Daten-Programm-Abhängigkeit**. Anwendungen, welche auf die Dateisystemstruktur basieren, enthalten meist eigene Dateiformate. Dies hat zur Folge, dass die Programme von den Dateiformaten abhängen. Wird der Aufbau einer Datei geändert, führt dies auch zu einer Veränderung aller darauf basierender Programme. Umgekehrt setzt eine Erweiterung der Funktionalität eines Programmes meist eine Veränderung des Dateiaufbaus beziehungsweise eine Restrukturierung der Datei voraus. Der Aufwand und die Schwierigkeit der Programmpflege verringern sich deutlich, wenn man strikt die Daten von den Programmen trennt und auf eine Datenbank auslagert.[39]

Die **Inflexibilität** der Daten ist der nächste Nachteil dieser Speichermethode, denn sie ist nicht anwendungsneutral. Gerade wenn Daten aus mehreren unterschiedlichen Anwendungen benötigt werden, ist ein Zusammenführen problematisch.[40] Denn jedes Programm besitzt seine eigene spezifische Speicherdatei mit unterschiedlichen Formaten und ein Datenaustausch untereinander bzw. ein Zusammenführen in anderen Programmen ist so gut wie unmöglich.

Diese Probleme können mittels des Einsatzes von Datenbanksystemen gelöst werden. Dazu wird die sogenannte **Datenintegration** angewandt. Der Kerngedanke dahinter ist, dass alle Anwendungen mit denselben Daten arbeiten, welche an einer zentralen Stelle verwaltet werden. Dieser Gesamtbestand an Daten wird nun als Datenbank bezeichnet. Dabei muss eine besondere Aufmerksamkeit auf den Entwurf solch einer Datenbank gelegt werden. Für das obige Beispiel mit dem

[39] Riccardi, Greg; Datenbanksysteme mit Internet- und Java-Applikationen; S. 36

[40] Schneider, Markus; Implementierungskonzepte für Datenbanksysteme; S.2

mehrfach verwendeten Datum bedeutet Datenintegration, dass das Datum nur einmal in der Datenbank abgespeichert wird und alle Anwendungen parallel darauf zugreifen können. Bei einer Veränderung muss nur dieser eine Datensatz korrigiert werden. Außerdem können moderne Datenbankmanagementsysteme Fragen wie Effizienz, Parallelität, Zugriffskontrolle und Datensicherheit zufriedenstellend beantworten.[41]

Es fielen in den letzten Absätzen vermehrt die Begriffe Datenbank, Datenbanksysteme und Datenbankmanagementsysteme. Da diese fälschlicherweise oft als Synonym verwendet werden, sollten diese kurz definiert und charakterisiert werden[42]:

Eine **Datenbank (DB)** ist eine zentrale Datensammlung, in der die Daten mit zugehörigen Beschreibungen abgelegt sind. Eine Datenbank kann dabei nur selten direkt verwendet werden. Man kann diese Daten meist mit einem Editor betrachten, diese würden aber keine verwertbaren Informationen enthalten. Vielmehr ist es eine reine, unsortierte Datensammlung. Zudem stößt man des Öfteren auf Zugriffsbeschränkungen, um eine Einsicht oder Manipulation auf empfindliche Daten durch Unbefugte zu erschweren.

An dieser Stelle greift das **Datenbankmanagementsystem (DBMS)** ein. Es ist eine Art Sammlung von Werkzeugen, welche in der Lage sind, Datenbanken zu verwalten. So können Daten schnell und effizient erstellt, bearbeitet und gelöscht werden. Das DBMS besitzt eine Benutzerschnittstelle, welche dem Anwender die Daten übersichtlich präsentiert und die Möglichkeit zur parallelen Verwaltung mehrerer Datenbanken gibt. Das DBMS kann also als Verwaltungssoftware für die Datenbank angesehen werden.

---

[41] Saake, Gunter; Sattler, Kai-Uwe; Heuer, Andreas; Datenbanken – Konzepte und Sprachen; S.4

[42] http://www.sql-und-xml.de/sql-tutorial/datenbank-grundbegriffe.html

Das **Datenbanksystem (DBS)** hingegen vereint in sich die Datenbank und Datenbankmanagementsystem. Es beschreibt somit den Oberbegriff über die Thematik. Die folgende Grafik stellt diese Abgrenzung noch einmal bildlich dar:

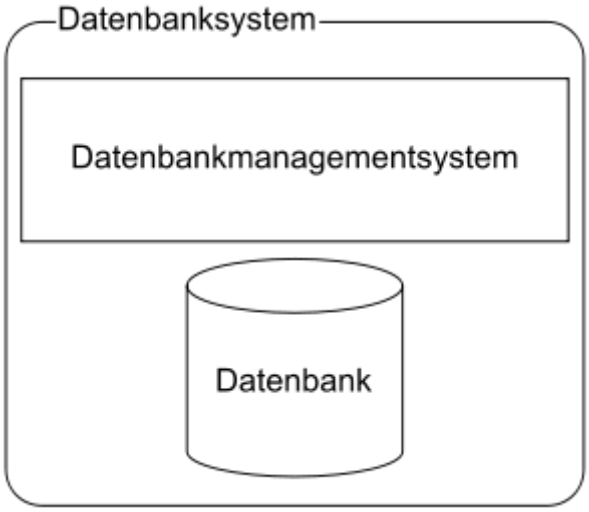


Abbildung 16 Datenbanksystem[43]

2.2.1 Datenbankmodelle

Bei der Entwicklung einer Datenbank ist es wichtig, sich vorher Gedanken über den Aufbau und die Modellierung zu machen. Eine Datenbank muss bereits in der Entwicklungsphase optimal auf die Anforderungen und an das Datenbankmanagementsystem angepasst werden. Nur so kann eine maximale Effektivität und eine optimale Nutzung garantiert werden. Daher wird in der Praxis der Realisierung eines Datenbanksystems eine Designphase voran gestellt. Dafür werden Modelle zu Hilfe gezogen. Modelle sind eine vereinfachte und abstrahierte Darstellung der Wirklichkeit[44]. In der Informationstechnologie nennt man diese Modelle **Datenbankmodelle**.

[43] http://upload.wikimedia.org/wikipedia/de/thumb/f/f3/Datenbanksystem.svg/685px-Datenbanksystem.svg.png

[44] Geisler, Frank; Datenbanken – Grundlagen und Design

Thomas Kudraß definiert ein Datenbankmodell dabei wie folgt:

> „Ein Datenbankmodell ist ein Datenmodell für ein Datenbanksystem. Es bestimmt, auf welche Art und Weise Daten prinzipiell gespeichert werden und wie man die Daten manipulieren kann.“[45]

Unter einem Datenbankmodell versteht man demnach eine abstrahierte Darstellung der Daten und den zwischen diesen Daten bestehenden Beziehungen. Es legt fest, auf welche Art und Weise die Daten in einem Datenbanksystem gespeichert werden.

In der Literatur wird klassisch in fünf grundlegende Datenmodelle unterschieden:

Hierarchisches Datenbankmodell

Das Hierarchische Datenbankmodell ist eines der ersten Modelle. Es ist, wie für hierarchische Modelle typisch, als Baumstruktur aufgebaut. An oberster Stelle steht der Parent-Datensatz, welche dann mehrere, ihm untergeordnete, Child-Datensätze beinhalten kann. In der dritten Ebene kann ein Child-Datensatz ebenfalls mehrere, sogenannte Blatt-Datensätze besitzen.

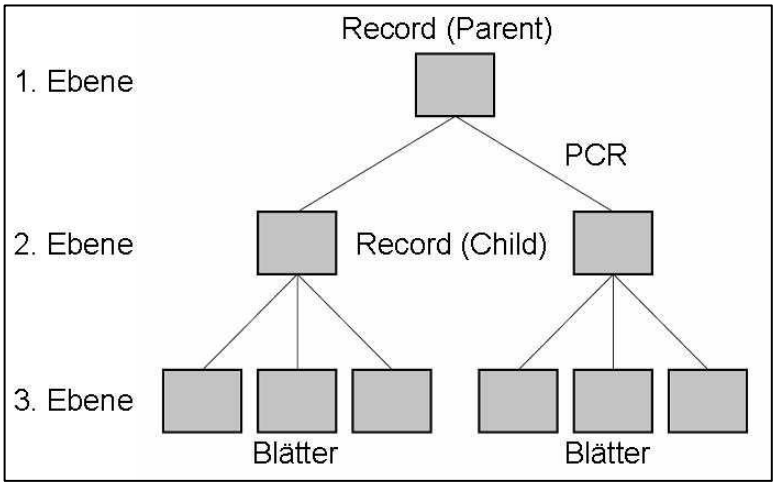


Abbildung 17 Hierarchisches Datenbankmodell[46]

[45] Kudraß, Thomas; Taschenbuch Datenbanken

[46] http://www.info-wsf.de/index.php/Datenbankmodelle#Hierarchisches_Datenbankmodell

Der Nachteil an diesem Datenmodell ist allerdings, dass es lediglich 1:1 bzw. 1:n-Beziehungen geben kann. Ein Parent-Datensatz kann also ein oder mehrere Child-Datensätze unter sich vereinen, während umgekehrt ein Child-Datensatz nur maximal einem Parent-Datensatz untergeordnet ist. Die Datenbanktypische n:m-Beziehung ist also in diesem Datenbankmodell nicht möglich.

Heute wird dieses Modell noch in Banken und Versicherungen eingesetzt, da es bei großen Datenmengen relativ kurze Zugriffszeiten gewährt.[47]

Netzwerk-Datenbankmodell

Dieses Datenbankmodell kann als eine Verallgemeinerung des Hierarchischen Datenbankmodells angesehen werden. Es besteht aus sogenannten Datasets, welche aus Owner und Member bestehen. Jeder Owner kann dabei keinen, einen oder mehrere Member besitzen. Die Beziehung wird dann mit dem Set-Type definiert. Am Beispiel der Beziehung Firma-Mitarbeiter könnte dies wie folgt aussehen:

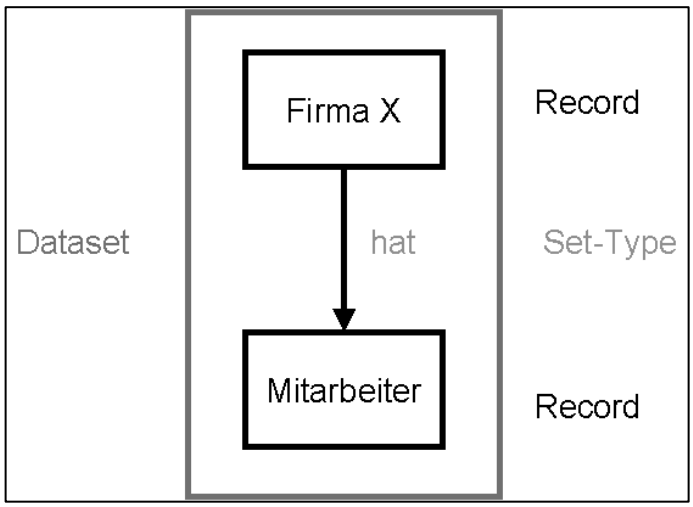


Abbildung 18 Netzwerk-Datenbankmodell[48]

[47] http://www.info-wsf.de/index.php/Datenbankmodelle#Hierarchisches_Datenbankmodell

[48] http://www.info-wsf.de/index.php/Datenbankmodelle#Netzwerkdatenbankmodell

Auch dieses Modell bietet den Vorteil der schnellen Zugriffszeiten. Es wird daher hauptsächlich auf Großrechnern eingesetzt, jedoch wird es immer öfter durch das relationale Datenbankmodell ersetzt.[49]

Relationales Datenbankmodell

In einer relationalen Datenbank werden die Daten in sogenannten Relationen dargestellt. Diese Relationen können als zweidimensionale Tabellen verstanden werden. Die Spalten dieser Tabellen stellen die Attribute (Eigenschaften) dar, während die Zeilen die Entitäten (Objekte) charakterisieren.[50] Im Normalfall stehen diese Tabellen in Beziehung zueinander. Also eine Tabelle verweist üblicherweise, über Schlüsseleinträge, auf andere Tabellen. Dabei sollte darauf geachtet werden, dass die Daten nicht redundant inkonsistent werden.

Im folgenden Beispiel ist eine relationale Datenbank zu sehen:

Tabelle 1 Nicht normalisierte relationale Datenbank - Gesamttabelle

Kunden-Nr.	Kunden-Name	PLZ	Ort	Artikel-Name	Preis pro Artikel	Anzahl
001	Meier	70038	Stuttgart	Schrauben	2	10
001	Meier	70038	Stuttgart	Nägel	2	8
001	Meier	70038	Stuttgart	Stahlblech	8	2
002	Schmidt	10115	Berlin	Schrauben	2	4
002	Schmidt	10115	Berlin	Nägel	2	9
003	Vogt	20095	Hamburg	Stahlbleche	8	7

An dieser Tabelle sieht man recht deutlich, dass viele Datensätze, wie z.B. die PLZ und Namen mehrfach vorkommen. Sollte der letzte Datensatz beispielsweise gelöscht werden, wären alle Informationen für Herrn Vogt aus der Datenbank entfernt. Sinnvoller wäre es daher, die Tabellen aufzuteilen. Denkbar wäre es, die Kunden- und Produktdaten auszulagern und über Schlüsseleinträge miteinander zu verknüpfen bzw. eine Relation herzustellen.

[49] http://www.info-wsf.de/index.php/Datenbankmodelle#Netzwerkdatenbankmodell

[50] Riccardi, Greg; Datenbanksysteme mit Internet- und Java-Applikationen; S. 113

Tabelle 2 Kundentabelle

Kunden-Nr.	Kunden-Name	PLZ	Ort
001	Meier	70038	Stuttgart
002	Schmidt	10115	Berlin
003	Vogt	20095	Hamburg

Tabelle 3 Produkttabelle

Artikel- Nr.	Artikel-Name	Preis
10021	Schrauben	2
10035	Nägel	2
10094	Stahlblech	8

Tabelle 4 Rechnungstabelle

Rechnungs-Nr.	Kunden-Nr.	Artikel-Nr.	Anzahl
2013-01	001	10021	10
2013-02	001	10035	8
2013-03	001	10094	2
2013-04	002	10021	4
2013-05	002	10035	9
2013-06	003	10094	7

Durch diese Aufteilung in drei einzelne Tabellen ist eine vereinfachtere Verwaltung gewährleistet. Änderungen am Kundenstamm können an einer einzigen Stelle getätigt werden und auch das Ändern des Produktpreises ist für alle gekauften Artikel konsistent. Jede einzelne Tabelle hat einen Schlüssel in Form einer Nummerierung. In der Rechnungstabelle wird dann auf einen bestimmten Kunden und einen bestimmen Artikel durch genau diese Schlüssel (Fremdschlüssel) verwiesen. Diesen Vorgang des Aufsplittens von Daten in mehreren Tabellen zur Vermeidung von Redundanzen nennt man **Normalisierung**.[51]

[51] Wieken, John-Harry; SQL - inkl. Lerntest auf CD: Einstieg für Anspruchsvolle; S. 243

Eine relationale Datenbank wird meist mittels eines Entity-Relationship-Modells (ERM) dargestellt und entworfen. Dieses besteht klassisch aus einer Grafik und einer zugehörigen Beschreibung. Dabei werden Objekte festgelegt, deren Beziehung untereinander definiert und Attribute bestimmt. Ein typisches ERM könnte so aussehen:

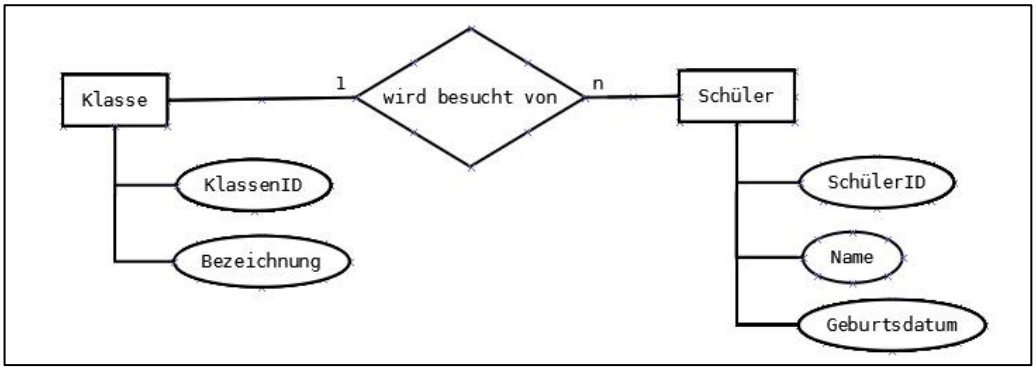


Abbildung 19 Entity-Relationship-Modell[52]

Das Entity-Relationship-Modell wird heute am häufigsten verwendet, da es sehr einfach darzustellen ist. Als Nachteil kann genannt werden, dass die Struktur durch die hohe Anzahl an Tabellen groß und intransparent werden kann und Informationen weit verstreut sind.

Objektorientiertes Datenbankmodel

Dieses Modell wurde entwickelt, um hochkomplexe Zusammenhänge, welche relationale Datenbanken schnell an die Grenze des Möglichen treiben, zu modellieren. Der Ansatz ist dabei, ähnlich wie bei der objektorientierten Programmierung (Kapitel 2.3.2), folgender: Es wird versucht die Realität in Form von Objekten zu beschreiben. Diese Objekte besitzen Methoden und Attribute, welche das Objekt näher beschreiben.

Das Objektorientierte Datenbankmodell bietet dabei folgende Vorteile[53]:

[52] http://www.datenbank-grundlagen.de/bilder/beispiel-Entity-Relationship-Modell.jpg

[53] Stocker, Christine; Objektorientierte Datenbanksysteme Inhalt, Bedeutung und Beispiel; S. 2

- In Objekten bilden die Methoden und die Attribute eine Einheit, während bei relationalen Datenbanken die Objekte oft in mehreren Datensätzen verteilt sind
- Objektmerkmale können auch komplex sein, was z.B. oft im Bibliotheksbereich gewünscht wird.
- Die Darstellung von Objekten samt Methoden und Attribute lassen sich einfacher und übersichtlicher visualisieren als eine Vielzahl von Tabellen und Datensätzen
- Objektorientierte Datenbanksysteme lassen sich gut mit objektorientierten Programmiersprachen wie Java und C++ koppeln.

Dieses Modell konnte sich allerdings, trotz guter Zugriffszeiten, bislang nicht durchsetzen.

Objektrelationales Datenbankmodell

Das Objektrelationale Datenbankmodell ist eine Mischform des relationalen und des objektorientierten Datenbankmodells. Der Gedanke dahinter ist, dass beide Datenbankmodelle viel gemeinsam haben. Auch in der relationalen Datenbank gibt es Objekte (meist Entitäten genannt) und Klassen (Entitätstypen). Das objektrelationale Datenbankmodell nutzt diesen Umstand, indem es die relationale Datenbank um objektorientierte Methoden und Datentypen erweitert. Somit sind auch komplex strukturierte Daten visualisierbar. In der Praxis findet dieses Datenbankmodell jedoch kaum Anwendung und ist noch in der Entwicklung.[54]

Zusammenfassung Datenbankmodelle

Nach der Analyse wird ersichtlich, dass es derzeit nur zwei gängige Datenbankmodelle gibt, welche auch praktische Bedeutung besitzen. Dies ist zum einen das objektorientierte Datenbankmodell, welches immer dann angewandt wird, wenn es sich um sehr komplexe Daten handelt. Jedoch existiert keine starke Anfragesprache für dieses Modell. Außerdem werden fortgeschrittene Programmierfähigkeiten vom Anwender vorausgesetzt, da es eng an objektorientierter Programmierung

[54] http://www.info-wsf.de/index.php/Datenbankmodelle#Objektorientiertes_Datenbankmodell

gekoppelt ist. Bei einfach strukturierten Datenbeständen ist das objektorientierte Datenbankmodell dem relationalen deutlich unterlegen. Es ist durch den Aufbau in Tabellen schnell zu verstehen und einfach anzuwenden. Die Abfragen sind mittels der Sprache SQL leicht anzuwenden. Bei komplexeren Daten kommt dieses Modell jedoch schnell an seine Grenzen. Für den herkömmlichen Gebrauch reicht die Funktionalität einer relationalen Datenbank allerdings meist aus und wird aus diesem Grunde auch in den meisten Fällen eingesetzt.

Eine gute Übersicht über die letzten drei Datenbankmodelle bietet folgende Tabelle[55]:

	Relationale Datenbanken	***Objektorientierte Datenbanken***	***Objektrelationale Datenbanken***
Standard	Altes, erprobtes, mathematisches Modell. Standardisiert.	Nicht vollständig vereinheitlicht.	In der Entwicklung
Handhabung von kleinen Objekten aus atomaren Typen	Bestens geeignet	Navigation meist zu umständlich	Gut, da relationales Modell vollständig integriert
Handhabung von Komplexen Daten	Benötigt intern viele Tabellen => Performance leidet.	Einfach möglich durch Konzepte der objektorientierten Programmierung	Möglich durch Bereitstellung benutzerdefinierbarer Datentypen.
Hierarchien	Nur sehr aufwendig modellierbar	Typhierarchien durch Vererbung	Typ- und Relationenhierarchien
Datenzugriff	Anfragesprache nicht-Prozedural	Einzelne Objekte Prozedurale Formulierung	Anfragesprache
Datensicherheit	Große Datenunabhängigkeit durch Drei-Schichten-Modell	Anfällig bei Typänderungen wegen Objektgebundenheit	Abhängig von verwendeten Datentypen
Anbindung an Objektorientierte Programmiersprache	Schwierig, „Impedance Missmatch“ tritt auf	Nähe zur Programmiersprache erleichtert Anbindung	Einfacher Datenaustausch

Abbildung 20 Gegenüberstellung Datenbankmodelle

[55] Spickermann, Gunnar ; pi.informatik.uni-siegen.de; Vergleich der Datenbankmodelle.pdf

2.2.2 Typische Vertreter

Heute gibt es auf dem Markt zahlreiche Datenbank-Produkte. Das wohl am weitesten Verbreitete ist die Oracle Database. Diese kann sowohl relationale als auch objektorientierte Daten verarbeiten. Ein starker Konkurrent für dieses System stellt das von Microsoft entwickelte MS SQL Server dar. Da Microsoft-Produkte in Unternehmen oft und gern verwendet werden und diese auch meist bestrebt sind, soviel Software wie möglich von einem Unternehmen zu nutzen, ist dieses relationale Datenbankmanagementsystem weit verbreitet. Der dritte große Mitspieler auf dem Markt ist der IT-Konzern IBM, welcher mit DB2 und Informix gleich zwei DBMS vertreibt. Es gab seitens IBM Überlegungen, diese beiden Systeme zusammenzuführen. Dies führte allerdings zu einem großen Unmut bei den Informix-Nutzern, woraufhin der Plan verworfen wurde und beide Systeme nun parallel weiterentwickelt und vertrieben werden.

Als Open-Source-Variante spielt noch MySQL eine große Rolle. Dieses relationale Datenbankmanagementsystem wurde von dem Unternehmen Sun Microsystems entwickelt. Da dieses Unternehmen allerdings 2010 von Oracle aufgekauft wurde, gehört MySQL nun ebenfalls zu Oracles Produktpalette.

2.3 Datenübertragung/Programmierung

Damit zwei Objekte miteinander kommunizieren können, werden sie über sogenannte Schnittstellen miteinander verbunden. Dadurch ist nun eine einfache, hardwaremäßige Verbindung hergestellt. Es muss jedoch ein Übertragungsprotokoll bzw. ein Algorithmus gefunden werden, welches die Kommunikation regelt. Das heißt, es muss ein Programm vorhanden sein, welches das Senden und Empfangen von Daten genau steuert. So wird bestimmt, auf welcher Art, zu wel-

cher Zeit und über welche Schnittstelle die nötigen Informationen ausgetauscht werden. Diese Anweisungen sind gerade bei solch einzigartigen Anwendungen, wie in dieser Arbeit beschrieben, äußerst speziell und benötigen daher in der Regel eine eigens angefertigte Programmierung.

2.3.1 Softwareentwicklung

Bei der Entwicklung von Programmen gibt es verschiedene Vorgehensweisen. Damit effizient programmiert werden kann, ist es hilfreich, gewisse organisatorische Rahmen festzulegen, welche eine genaue Planung der einzelnen Schritte und Aufgaben genauer definieren. Gerade bei größeren Projekten ist dies unabdingbar. Ein unstrukturiertes Programmieren ist in den seltensten Fällen erfolgversprechend. Solch organisatorische Rahmen werden Prozessmodelle genannt. Diese Modelle beantworten folgende Fragestellungen:[56]

- Welche Phasen und Reihenfolge gibt es im Arbeitsablauf?
- Welche Aktivitäten sollen durchgeführt werden?
- Gibt es Verantwortlichkeiten, welche zugeteilt werden müssen?
- Was sind die erforderlichen Inputs?
- Welche Standards, Richtlinien, Methoden und Werkzeuge sollen angewandt werden?

Dabei wird zwischen zwei Hauptgruppen unterschieden:

- Die klassischen Methoden
- Die agilen Methoden

In **klassischen Vorgangsmodellen** werden die Projektphasen eher sequentiell abgehandelt. Die Projekte werden in fest definierte Abschnitte unterteilt, welche bestimmte Ergebnisse erbringen sollen. Erst wenn diese Ergebnisse vorliegen, ist der Teilprozess beendet und es wird mit dem nächsten begonnen. Diese Methoden beschreiben, wie etwas während eines Projekts gemacht werden soll und vor

[56] http://www.scrum-kompakt.de/grundlagen-des-projektmanagements/vorgehensmodelle-in-der-softwareentwicklung/

allem, was genau in welchem Dokument zu einer Zeit beschrieben sein muss, sodass Software entwickelt werden kann. Allerdings hat dies den Nachteil, dass ein Kunde erst sehr spät einen Einblick in das Produkt bekommt und somit eventuelle Fehlentwicklungen zu spät erkannt werden. Ein zwischenzeitliches Abgleichen ist nur schwer möglich. Dies versucht man durch vorgegebene Richtlinien und Vereinbarungen (Pflichten- und Lastenheft) zu vermeiden. Zusammengefasst kann man sagen, dass klassische Methoden versuchen, schon von Beginn an eine maximale Planbarkeit des Projektes zu gewährleisten. Typische Vertreter für diese Methoden sind das Wasserfall- und Spiralmodell, sowie „Rational Unified Process“ (RUP) und das VModell.

Typische Schritte eines Wasserfallmodells nach Royce sehen Beispielsweise so aus:

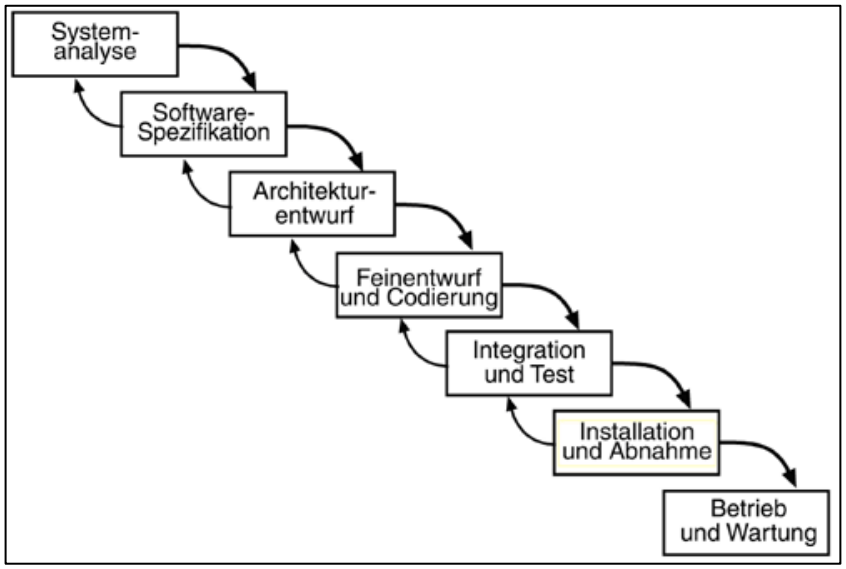


Abbildung 21 Wasserfallmodell als Vertreter für klassische Vorgangsmodelle[57]

Die Projektphasen sind strikt voneinander abgegrenzt und müssen nacheinander abgearbeitet werden. Erst wenn eine Phase beendet ist, kann zur nächsten übergegangen werden.

Gerade an diesem Wasserfallmodell zeigen sich die Nachteile deutlich: Anforderungen und Leistungen müssen schon zu Beginn des Projektes präzise be-

[57] http://www.scrum-kompakt.de/files/2010/04/wasserfall-modell-royce.gif

schreibbar sein. Änderungen in den Anforderungen führen entweder zu einem Produkt, welche diese nicht erfüllt oder das Projekt muss umgestoßen und neu begonnen werden. Diese Inflexibilität ist nicht gewollt. Damit man in der heutigen Wirtschaft auf Dauer auf dem Markt überlebensfähig bleibt, ist es nötig, zügig auf Änderungen reagieren zu können. Daher entstanden die agilen Methoden.

Das **agile Vorgangsmodell** soll die Möglichkeit bieten, jederzeit flexibel auf Änderungen der Rahmenbedingungen reagieren zu können. Im Gegensatz zu den klassischen Modellen wird hier nicht mehr an feste Phasen und Strukturen festgehalten, sondern es wird versucht, mit dem Kunden zusammen im Dialog das Produkt zu entwickeln. Der Kunde bekommt also was er braucht, nicht was er spezifiziert hat. Die Spezifikationen werden im Laufe des Projekts kontinuierlich erweitert und bearbeitet. Der Auftraggeber kann so während des kompletten Prozesses Einfluss nehmen. Es wird zudem versucht, den bürokratischen Aufwand so gering wie möglich zu halten.

Typische Vertreter für dieses Vorgangsmodell sind das „Extreme Programming (XP)“ und Scrum.

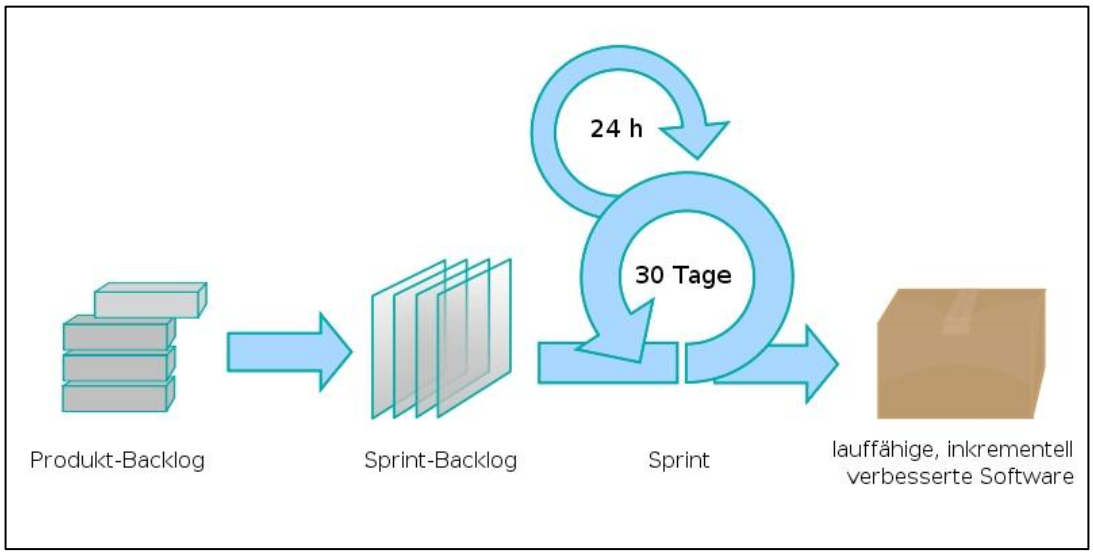


Abbildung 22 Scrum-Vorgangsmodell[58]

[58] http://www.techdivision.com/blog/agile-projektentwicklung-mit-scrum/

Nachdem nun die Prozessmodelle näher beschrieben wurden, kann sich intensiver mit dem einzelnen Projekt beschäftigt werden. Jeder Programmierer muss sich dabei für eine geeignete Programmiersprache entscheiden. Diese Entscheidung beruht allerdings auf einem zuvor definierten Programmierstil. Dem sogenannten Programmierparadigma.

2.3.2 Programmierparadigma

Ein Programmierparadigma kann als das jeder Programmiersprache zugrunde liegende Prinzip bezeichnet werden. Dabei wird in der Fachliteratur zwischen imperativen (lat. imperare = anordnen, befehlen) und deklarativen (lat. declarare = erklären) Paradigmen unterschieden. Jede dieser beiden Untergruppen trifft noch einmal spezifische Unterteilungen. Erwähnenswert ist dabei die Tatsache, dass jede Programmiersprache mehrere Programmierparadigmen beinhalten kann. Eine mögliche Unterteilung sieht z.B. so aus:

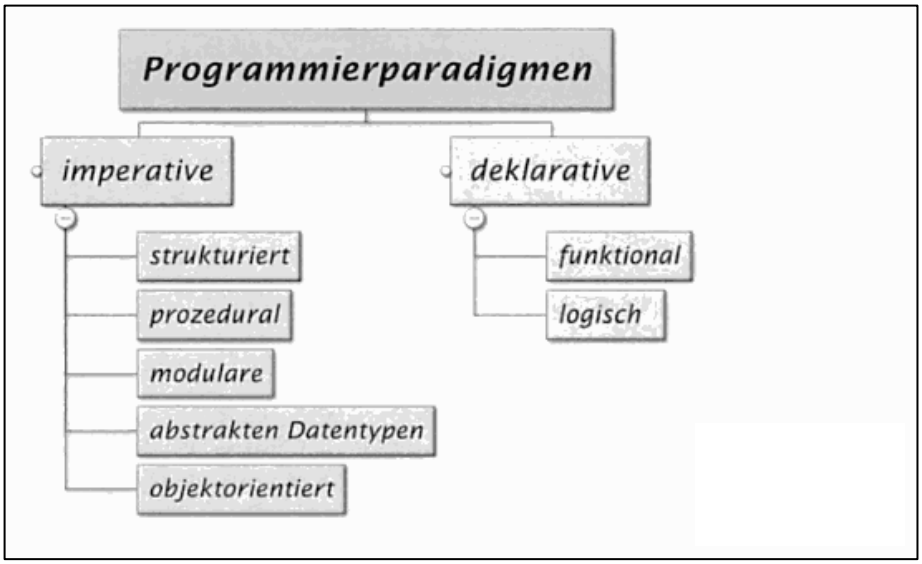


Abbildung 23 Programmierparadigmen[59]

Bei imperativen Programmierparadigmen werden einzelne Anweisungen als lineare Folge von Befehlen angesehen. Diese Befehle werden dann in einer definierten Reihenfolge von dem auszuführenden System abgearbeitet.

[59] Kannengiesser, Matthias; Objektorientierte Programmierung mit PHP; S. 33

Im Gegensatz dazu verfolgen die deklarativen Programmierparadigmen den Ansatz, dass direkt auf die Problemlösung hin programmiert wird und nicht der Lösungsweg. Es steht also das „Was“ im Vordergrund und nicht das „Wie“. Oft beruhen diese Paradigmen dann auf mathematische und rechnerunabhängige Theorien.[60]

2.3.3 Programmiersprachen

Jeder Programmiersprache liegen eine oder mehrere Programmierparadigmen zu Grunde. Die Auswahl einer Sprache ist jedoch nicht nur vom gewünschten Paradigma abhängig, sondern auch von einer Vielzahl anderer Faktoren. Das Zielsystem, die weitere Software, die verwendete Hardware und die weitere Pflege des Programms sind nur einige Kriterien für die Sprachauswahl. Jede der zahlreichen Programmiersprachen hat ihre eigenen Vor- und Nachteile.

Im Anschluss werden nun die für diese Arbeit in Frage kommenden Programmiersprachen eingeführt. Selbstverständlich gibt es noch zahlreiche Weitere. Auf deren Beschreibung wird allerdings verzichtet, da nur diese drei im Verlauf der Arbeit zur Auswahl stehen werden.

C++

Die von Bjarne Stroustrup entwickelte Programmiersprache C++ ist eine Weiterentwicklung und Erweiterung von C um eine Objektorientierung und wurde zunächst als „C mit Klassen“ bezeichnet.[61] C++ ist dabei keine reine objektorientierte Sprache, sondern vielmehr eine Hybridsprache. Somit ist auch eine Kompatibilität zu C gewährleistet.

Objektorientiert bedeutet dabei, dass Daten in Objekte zusammengefasst werden und diese Methoden und Eigenschaften besitzen. Dies führt zu einem modularen Aufbau, welcher leichter zu lesen ist. Auch der Aufbau von Programmen wird dadurch deutlich erleichtert.

---

[60] Kannengiesser, Matthias; Objektorientierte Programmierung mit PHP; S. 35

[61] Stroustrup, Bjarne; Einführung in die Programmierung mit C++; S. 28

Sie wurde um zusätzliche Sprachelemente erweitert. So sind in C++ Referenzen, Templates und Ausnahmebehandlungen (Exception Handling) integriert. Diese haben zwar keinen direkten Bezug zur Objektorientierung, sind aber für deren effiziente Implementierung wichtig.[62]

C++ wird sowohl für die Systemprogrammierungen als auch für die Anwendungsprogrammierungen eingesetzt und zählt unter Entwicklern zu einer der am weitesten verbreiteten Programmiersprachen.

Java

Java ist eine objektorientierte Programmiersprache, welche 1995 von der Firma Sun Microsystems entwickelt wurde. Sie ist nach Angaben des Herstellers auf rund 850 Millionen PCs und Millionen anderen Geräten, wie Mobiltelefonen und TV-Geräten installiert.[63]

Der größte Vorteil von Java ist seine Plattformunabhängigkeit. Das bedeutet, dass die erstellten Programme ohne Probleme auf jedem Endgerät mit installierter Laufzeitumgebung lauffähig sind. Möglich ist dies, da der vom Programmierer eingegebene Quellcode nicht direkt ausführbar ist, sondern erst durch einen Java-Compiler in den sogenannten Java-Bytecode übersetzt wird. Dieser Code wird dann allerdings nicht von einer Hardware ausgeführt sondern auf dem Zielsystem von einer entsprechenden Software (virtuelle Maschine). Somit ist keine Abhängigkeit mehr von der Hardware vorhanden und die Programme sind auf allen Systemen identisch lauffähig.

Weitere Vorteile[64] von Java sind unter anderem:

- objektorientiert
- hohe Robustheit
- unterstützt multi-threading (parallele Aktionen, die das Programm gleichzeitig ausführen kann)

---

[62] Kirch, Ulla; Prinz Peter; C++ - Lernen und professionell anwenden, S. 23

[63] Vgl. http://www.java.com/de/download/faq/whatis_java.xml

[64] http://ddi.cs.uni-potsdam.de/HyFISCH/Produzieren/SeminarDidaktik/OOP/java_vor_nachteile.html

- ist eng mit C++ Verwandt
- hohe Portabilität

Der größte Nachteil an Java ist die verringerte Performance, die aus der Plattformunabhängigkeit resultiert. Dies macht Java im Vergleich zu Anwendungen in C++ etwas langsamer.

JavaServer Pages (JSP)

JavaServer Pages ist eine Antwort der Firma SUN auf Microsofts ASP (Active Server Pages) und ist eine serverseitige Programmiersprache. Bei einer JavaServer Page handelt es sich im Grunde um eine Datei, bestehend aus einem HTML-Grundgerüst, welche mit Java-Code ergänzt wird. Diese Java-Codes werden Scripting-Elemente genannt und sorgen für eine hohe Dynamik der an sich recht statischen HTML-Dateien.[65]

Zum Ausführen einer solchen JSP wird ein Applikationsserver, wie z.B. Apache Tomcat benötigt. In diesem wird die JSP in Java-Quellcode (Java-Servlets) umgewandelt. Anschließend wird dieser in Bytecode konvertiert und kann auf einer virtuellen Maschine eines Webservers ausgeführt werden.

Auch hier kann, dank der Java-Technologie, der Vorteil der Plattformunabhängigkeit genannt werden. Wurde also beispielsweise eine JSP-Datei unter Linux erstellt, kann diese unter allen anderen Betriebssystemen ausgeführt werden.[66] Dies ist von besonderer Bedeutung, wenn im System mehrere Betriebssysteme verwendet werden sollen.

Man sollte jedoch JSP von dem meist geläufigeren JavaScript unterscheiden. JSP’s werden auf Servern verwendet, während JavaScript eine Skriptsprache ist, welche direkt beim Anwender wirksam wird. Man kann also sagen, dass JSP den Inhalt der kompletten Website beeinflusst, während JavaScript die Präsentation auf dem Bildschirm regelt.

---

[65] Balzert, Helmut; JSP für Einsteiger, S.13

[66] JSP mit Tomcat, Michael Seeboerger-Weichselbaum, S.22

3 Analyse des Ist-Zustandes der Firma Miele & Cie. KG

3.1 Qualitätsmanagement bei Miele

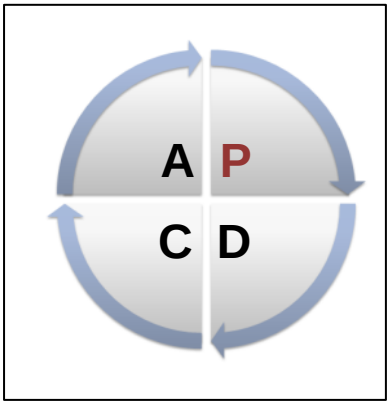


Abbildung 24
Gliederungspunkt Plan

In der Vergangenheit gab es bei Miele für das Qualitätsmanagement kein einheitliches Vorgehen und die Kompetenzen und Verantwortlichkeiten waren nicht durchgängig definiert. Außerdem haben alle Standorte ihre eigenen Qualitätsmaßnahmen durchgeführt. Ein genormtes, unternehmensweites Verfahren fehlte. Dadurch konnten keine einheitlichen Kennzahlen erhoben werden und eine Vergleichbarkeit war nicht gewährleistet. Auch das Wissen der Mitarbeiter über vorhandene QM-Instrumente war teilweise nicht vorhanden. Zwar fanden Schulungen statt, aber deren Wirksamkeit wurde aufgrund mangelnder konkreter Anwendung bezweifelt.[67]

Aus diesem Grunde wurde Anfang 2007 ein neues Projekt namens „Q-Kernprozess“ ins Leben gerufen. Ziel war es, die bewährte Miele-Qualität zu sichern und auf ein neues Niveau zu heben. Dazu wurden alle qualitätsrelevanten Prozesse neu ausgerichtet. Das neue Kernprozessmodell erlaubt nun die Sicherstellung:

- der Qualität mit Hilfe klar definierter Prozesse und Methoden,
- der Kaufteilqualität durch Lieferantensteuerung und –entwicklung,
- einheitlicher Prozesse und Methoden durch festgelegte Standards,
- der Qualitätsziele aus Kunden-, Entwicklungs-, Produktions- und Einkaufssicht durch einheitliche Q-Kennzahlen.

[67] Miele Intranet: http://mielewss.de.miele.net/wss/gtg/intranet/startseite/gtg_qm/qm_kernprozesse

Den Aufbau des kompletten Qualitäts-Kernprozess-Modells kann man folgender Grafik entnehmen:

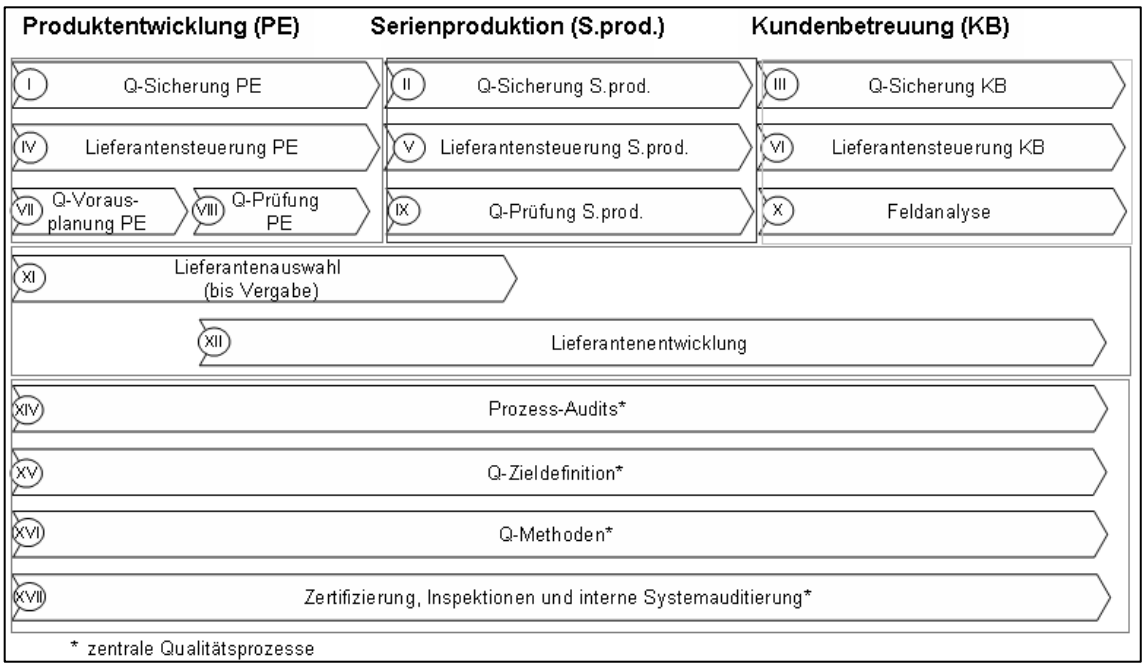


Abbildung 25 Kernprozessmodell

Das Modell kann demnach in fünf Hauptgruppen zerlegt werden. In Qualitätsmaßnahmen für:

- die Produktentwicklung,
- die Serienproduktion,
- die Kundenbetreuung,
- die Lieferanten und
- das zentrale Qualitätsmanagement.

Für die Sicherstellung der Qualität wurden 17 Methoden ausgewählt, welche nun zum Einsatz kommen. In der nachfolgenden Abbildung sind diese genannt und den jeweiligen Bereichen zugeordnet.

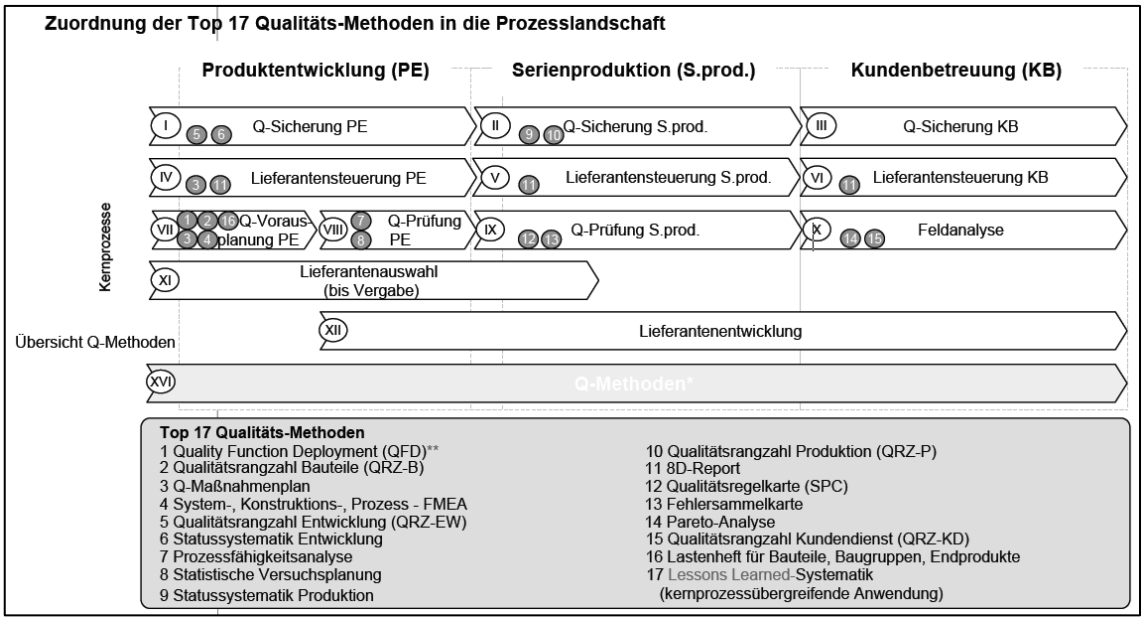


Abbildung 26 Qualitätsmethoden und deren Einsatzgebiet

Die vorliegende Diplomarbeit beschäftigt sich mit der Prüfung der produzierten Geräte, welche entweder noch in der Entwicklungsphase sind oder bereits in die Serienfertigung überführt wurden. Daher wird auf eine nähere Beschreibung der Kunden- und Lieferantenqualität, sowie des zentralen Qualitätsmanagements verzichtet.

3.1.1 Produktentwicklung

Schon während der Entwicklung eines Produktes muss auf eine bestimmte vorgegebene Qualität hin gearbeitet werden. Es muss sich im Voraus darauf geeinigt werden, welche Qualität das zukünftige Gerät erreichen soll. Während der Entwicklungsphase müssen alle Bauteile, Prototypen und Null-Serienteile verschiedene Prüfungen (Erstmusterprüfung, Lebensdauerversuche, Feldversuche) durchlaufen.

Bei dem Kernprozessmodell der Produktentwicklungsqualität gibt es vier Teilaufgaben, welche beachtet werden. Zunächst gibt es eine **Qualitätssicherung**, welche auf Qualitätsabweichungen reagiert. Es werden die Abweichungen aufgenommen, Ursachen analysiert, Gegenmaßnahmen eingeleitet und deren Wirksamkeit geprüft. Im zweiten Schritt werden die **Lieferanten** hinsichtlich der Quali-

tät der Materialien/Bauteile, der Einhaltung von vereinbarten Lieferzeiten und der Kosten untersucht. Außerdem wird eine **Qualitätsvorausplanung** durchgeführt, welche sich mit der präventiven Absicherung der vorgegebenen Zielqualität der Produkte und Prozesse beschäftigt. Dabei werden schwerpunktmäßig Risikoanalysen durchgeführt und Prüfpläne für die Produktentwicklung und Serienproduktion erstellt. Der letzte Punkt ist die **Qualitätsprüfung**, welche das Ziel verfolgt, eine unbegrenzte Serienfreigabe für das neue Produkt zu erhalten und somit den Start der Produktion einzuleiten.

3.1.2 Serienproduktion

Auch während der Serienfertigung muss die Qualität durchgehend im Auge behalten werden. Es ist nicht ausreichend, diese nur in der Entwicklungsphase zu untersuchen und dann davon auszugehen, dass die Qualität immer auf einem gleichhohen Niveau bleibt.

Das Vorgehen und die verwendeten Methoden ähneln dabei sehr denen der Produktentwicklung. In der Serienproduktion kann jedoch auf die Qualitätsvorausplanung verzichtet werden.

In diesem Bereich kommen jedoch andere Methoden zur Anwendung, welche der Abbildung 26 entnommen werden können.

Das Q-Kernprozessmodell führte zu einer gestiegenen Qualität und somit zu einer geringeren Ausfallrate der Geräte. Dadurch konnten die Kosten für Garantie- bzw. Kulanzleistungen gesenkt werden. Seit Mitte 2007 läuft das Projekt erfolgreich und wird an allen Standorten Schritt für Schritt eingeführt.

3.2 Prüfplätze

Die Geräte und Gerätekomponenten, Prüfobjekte genannt, werden auf ihre Langlebigkeit und dem Waschverhalten in Abhängigkeit der Zeit getestet. Insgesamt stehen dafür am Standort Gütersloh 189 Prüfplätze zur Verfügung. Dabei laufen

die Geräte unter Alltagsbedingungen, je nach Prüfprogramm, alle Programmschritte wiederholt durch und alle relevanten Werte werden dabei dokumentiert und kontrolliert. Abbildung 27 zeigt beispielhaft zwei der Dauerprüfplätze:

Abbildung 27 Dauerprüfplätze

Jeder einzelne Prüfplatz besitzt eine Rechnereinheit bestehend aus einem 200 MHz getakteten PowerPc-CPU, 64 Mb Hauptspeicher und 4Gb Flashspeicher, sowie analoge und digitale Schnittstellen.

Die Prüfobjekte sind direkt mit dem Prüfplatz über eine, von Miele entwickelte, optische Schnittstelle verbunden. Über diese Schnittstelle können Daten aus dem Prüfobjekt ausgelesen und bei Bedarf auch geschrieben werden. Dies ist für den technischen Kundendienst bei Softwarefehler ebenso wichtig wie für die betriebsinterne Geräteprüfung. Aufgrund einer Veränderung innerhalb der neuen Produktlinie wird derzeit eine verbesserte optische Schnittstelle eingeführt. Diese wird eine bedeutend höhere Übertragungsgeschwindigkeit erreichen.

Das Speichern der Prüfdaten sowie das Lesen der benötigten Grunddaten geschieht mittels einer Informix-Datenbank.

Jeder Prüfplatz besitzt zudem eine Mensch-Maschine-Schnittstelle in Form eines kleinen Displays und Bedienelementen wie in Abbildung 28 zu sehen. Diese bietet Informationen über die aktuelle Prüfung wie zum Beispiel die Laufzeit, Drehzahl und Soll- bzw. Ist-Zyklenzahl.

Abbildung 28 Mensch-Maschine-Schnittstelle

3.3 Prüfaufträge

Damit ein Prüfplatz seine Arbeit aufnehmen kann, muss vorher ein Prüfauftrag erstellt werden. Dieser Auftrag enthält sämtliche Daten, welche für die Durchführung nötig sind wie beispielsweise die Prüfaufgabe und einen Zieltermin. Erstellt wird dieser durch den Auftraggeber (z.B. Labore) mittels der Anwendungssoftware ProLab.

Ein Prüfauftrag besteht aus mindestens einem Prüfobjekt (Gerät, Gerätekomponente), welches wiederrum eine oder mehrere Prüfreihen zugewiesen bekommt. Diese Prüfreihen können weiterhin in mehreren Prüfungen unterteilt werden. Dieser hierarchische Aufbau kann grafisch wie folgt dargestellt werden:

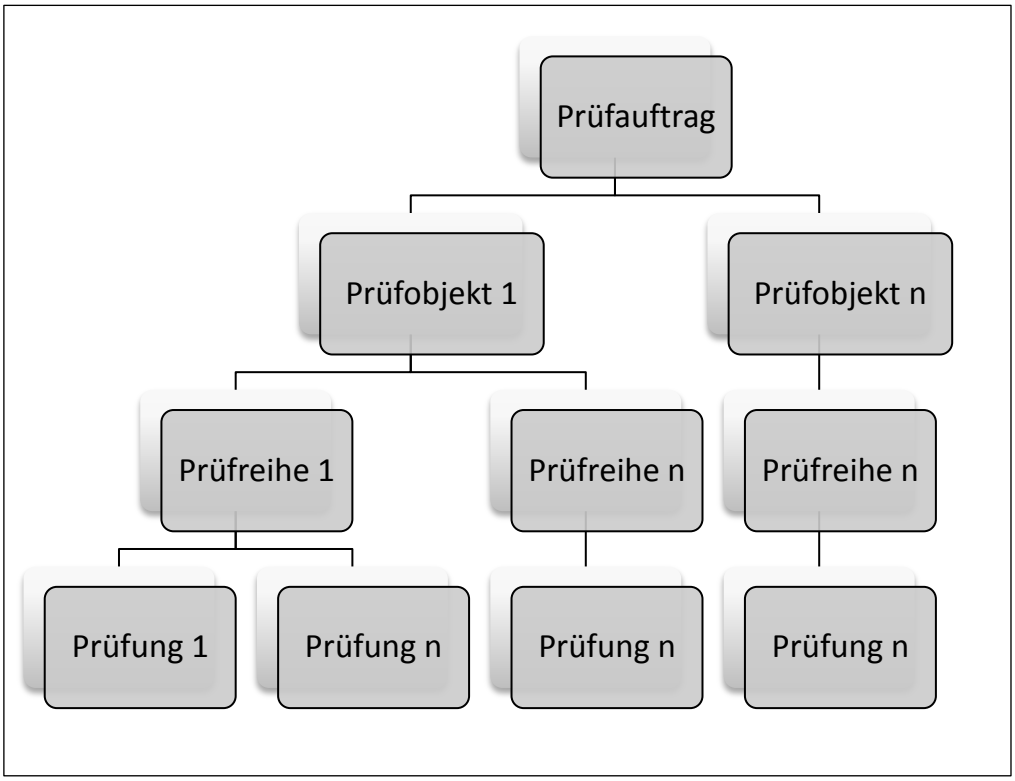


Abbildung 29 Aufbau Prüfauftrag[68]

Zur besseren Verdeutlichung folgt ein vereinfachtes Beispiel:

Es wird ein Prüfauftrag erstellt, welcher den Wasserverbrauch und die Wasserqualität zweier Waschmaschinen prüfen soll. Teil des Prüfauftrags sind also zwei Prüfobjekte Waschmaschine 1 und Waschmaschine 2.

Geprüft wird der Wasserverbrauch pro Waschmaschine. Um aufschlussreiche Ergebnisse zu erhalten, ist es nötig, diese Prüfung mehrmals durchzuführen, um Messfehler ausschließen zu können. Diese Einzelprüfungen werden zur Prüfreihe zusammengefasst. Parallel dazu wird in einer weiteren Prüfreihe die Wasserqualität vor und nach den Waschzyklen getestet.

[68] Anlehnung: Alexander Kuhn ; Bachelor-Arbeit: Entwicklung einer Applikation zur Erfassung und Ausgabe von Titrationsmessdaten als webbasierter Workflow; Miele & Cie. KG; S. 6

3.4 Anstehende Softwareveränderungen

3.4.1 Ablösung DFVDB durch ProLab/LabDB

Die Prüfplätze kommunizieren derzeit mit der Datenbank über die Dauerfeldversuchsdatenbank (DFVDB). Dieses basiert auf dem Datenbanksystem Informix und der Entwicklungsumgebung Informix New Era, welche auf Windows 2000 Servern lauffähig ist. Da Informix New Era nicht mehr vom Hersteller IBM unterstützt und weiterentwickelt wird, wurden weitere Anpassungen mittels Java-Programmierung vorgenommen. Die Anwendungsebene wurde als Eigenentwicklung bei dem Unternehmen Miele erstellt.

Die DFVDB ist bis heute im täglichen Einsatz. Es wird jedoch derzeit bei Miele an der Einführung eines neuen Labordatenmanagementsystems gearbeitet. Bei diesem handelt es sich um die Labordatenbank (LabDB) im Zusammenspiel mit dem Anwendungssystem ProLab.

Die **LabDB** ist eine von der Firma Werum entwickelte Datenbankanwendung, welches auf deren Produkt HyperTest basiert. Es wurde auf die speziellen Anforderungen von der Firma Miele angepasst und deckt die komplette Verwaltung der Prüfungen ab. Das HyperTest-System kommt bereits in vielen Bereichen der Automobil- und Luftfahrtindustrie zum Einsatz und der Arbeitsbereich beinhaltet dabei unter anderem die Verwaltung von Prüfaufträgen, Prüfreihen und Prüfobjekten.[69]

Die LabDB und die ihr zugrunde liegende HyperTest-Software wurde komplett mit Java umgesetzt und verwendet ein, von Oracle entwickeltes, relationales Datenbankmanagementsystem.

Als Anwendungssoftware dient das Programm **ProLab** (*Pro*vo en *Lab*oratorio, Prüfungen im Labor). Diese wird direkt vom Prüfer am PC bedient und unterstützt ihn bei der Durchführung der Prüfungen. ProLab bietet dem Labormitarbeiter neben Informationen zur Prüfung auch die Möglichkeit zum Einbinden von Messwer-

[69] http://www.werum.de/de/mdm/refer/index.jsp

ten und Ergebnissen. Die Anwendung dient also zum einfachen und effizienten Anlegen und Durchführen von Prüfungen. Es unterstützt den Anwender sowohl bei der Erstellung von Fehlermeldungen und Zwischenberichten als auch bei der Überwachung von laufenden Prüfungen.

ProLab ist eine, von Miele & Cie. KG selbst geschriebene, webbasierte Anwendung, bei der Java-Server-Pages (JSP) eingesetzt werden. Das Starten erfolgt direkt im Browser durch Aufrufen der JSP, ähnlich wie bei einer Website. Die Anwendung ist dabei modular aufgebaut. Als Application-Server wird, wie auch bei der zur Kommunikation nötigen DBAccess.jsp, ein Apache Tomcat-Server verwendet. Die Anwendung benötigt aufgrund seines Aufbaues lediglich die Laufzeitumgebung „XULRunner“ (XUL = XML User Interface Language).

Die Kommunikation zwischen der LabDB und ProLab erfolgt mit Hilfe des Java Messaging System (JMS). Der JMS-Provider ist der JBoss-Application-Server der Firma Redhat. Eine Informix-Datenbank dient ProLab als Datenbank.[70]

Der große Vorteil des neuen Systems, im Gegensatz zu dem älteren DFVDB-Systems, ist der flexible Einsatz innerhalb der verschiedenen Entwicklungslabore. Mit ProLab können nicht nur Prüfungen im Dauerversuch, sondern auch in anderen Laboren mit automatisierter Prüftechnik innerhalb der gleichen Anwendung durchgeführt werden.

Um LabDB und ProLab besser unterscheiden zu können und die Arbeitsaufgaben jeweils abzugrenzen, folgt eine Grafik:

[70] BA_AKuhn ProLab LabDB.pdf

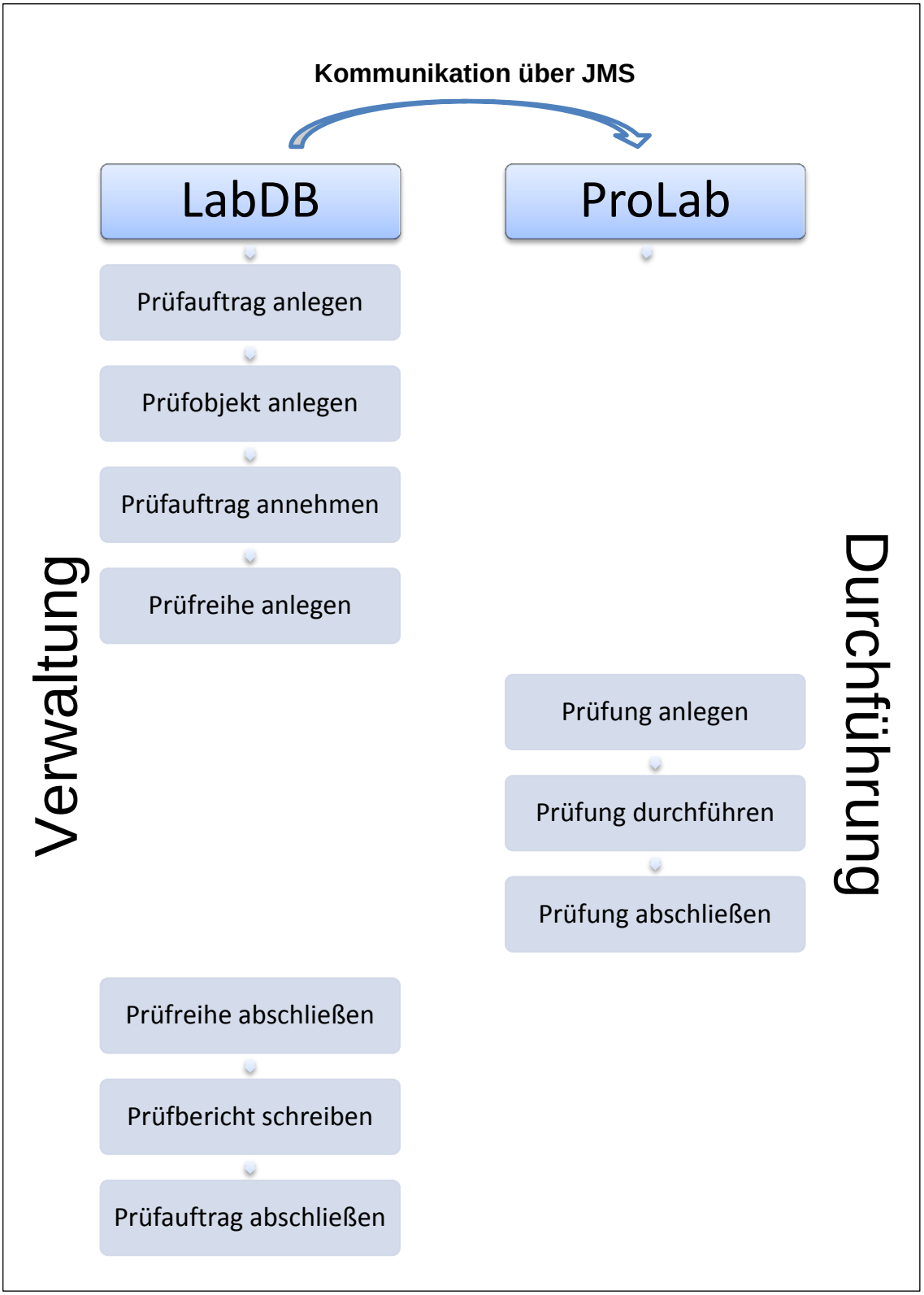


Abbildung 30 Aufgabenverteilung LabDB - ProLab[71]

[71] Anlehnung - LabDB - Die Miele-Labordatenbank - Allgemein.ppt – Miele Intranet

3.4.2 Ablösung von RTOS-UH durch QNX

Auch innerhalb der Prüfplätze gibt es eine softwareseitige Veränderung. Es wird derzeit an einer Umstellung des Betriebssystems von RTOS-UH auf QNX Neutrino der Firma QNX Software Systems gearbeitet.

RTOS-UH

RTOS-UH ist ein Akronym für **R**eal**T**ime**O**perating**S**ystem-**U**niversität**H**annover. Es ist demnach ein Echtzeitbetriebssystem und wurde speziell für die Anforderungen der Mess-, Steuer- und Regelungstechnik entworfen.

In der DIN 44300 wird Echtzeit wie folgt Definiert:

> „Unter Echtzeit versteht man den Betrieb eines Rechensystems, bei dem Programme zur Verarbeitung anfallender Daten ständig betriebsbereit sind, derart, dass die Verarbeitungsergebnisse innerhalb einer vorgegebenen Zeitspanne verfügbar sind. Die Daten können je nach Anwendungsfall nach einer zeitlich zufälligen Verteilung oder zu vorherbestimmten Zeitpunkten anfallen.“[72]

Der große Vorteil eines Echtzeitsystems ist es, dass man eine genaue Aussage darüber treffen kann, wie lange ein Prozess dauern wird und wann auf einen anderen umgeschaltet wird.[73]

Ein solches echtzeitfähiges Betriebssystem ist besonders interessant für die Prüftechnik. RTOS-UH unterstützt die Programmiersprache PEARL (Process and Experiment Automation Realtime Language), welche auch bei Miele genutzt wird. So ist es auch möglich, einzelne Lösungen in Teilprogrammen, den sogenannten Tasks, zu zerlegen und parallel auszuführen (Multitasking).[74]

---

[72] Scholz, Peter; Softwareentwicklung eingebetteter Systeme: Grundlagen, Modellierung, Qualitätssicherung, 2005, Springer Verlag S. 39

[73] Benra, Juliana T.; Softwareentwicklung für Echtzeitsysteme, 2009, Springer Verlag, S.68

[74] http://www.iep.de/Downloads/deutsch/RTOSstart.pdf

Die Universität Hannover bewirbt ihr Betriebssystem dabei unter anderem mit folgenden Vorteilen[75]:

- Kompakt
- Skalierbar
- Klare und präzise Architektur
- Echtzeitfähigkeit

Dieses Betriebssystem ist derzeit bei der Firma Miele auf allen Prüfplätzen installiert und im täglichen Einsatz. Allerdings wird es zukünftig nicht mehr weiter entwickelt und durch das Betriebssystem QNX ersetzt.

QNX

QNX ist ebenfalls ein Echtzeitbetriebssystem und basiert auf dem Unix-System. Es wurde erstmals 1982 von Gordon Bell und Dan Dodge vorerst unter dem Namen QUNIX veröffentlicht. 2001 wurde das System komplett überarbeitet und als QNX Neutrino vorgestellt.[76] Im Jahre 2009 wurde das Betriebssystem vom BlackBerry-Hersteller Research in Motion (RIM) gekauft.[77]

Folgende Argumente sprechen für QNX als Betriebssystem:

- Microkernelarchitektur
- Posix-Konformität (POSIX = Portable Operating System Interface)
- Multiprozessorfähigkeit
- Eigene grafische Oberfläche (Photon microGUI)
- Hardwareunterstützung (einige Treiber bereits fertig programmiert)[78]

[75] http://www.irt.uni-hannover.de/rtos/

[76] Vgl. http://heineck.hof-university.de/fileadmin/Dozenten/Horst_Heineck/PDF/Echtzeitsysteme/QNX.pdf, S. 4 und 5

[77] Vgl. http://press.rim.com/newsroom/press/2010/pressrelease-3766.html

[78] http://heineck.hof-university.de/fileadmin/Dozenten/Horst_Heineck/PDF/Echtzeitsysteme/QNX.pdf

Bei QNX handelt es sich um ein POSIX-kompatibles Echtzeitbetriebssystem, in dem Prozesse in geschützten Speicherbereichen ausgeführt werden und dadurch vor dem Zugriff von anderen Prozessen geschützt sind. Diese Eigenschaft ist ein weiterer wichtiger Grund für die Umstellung des Betriebssystems, denn in der aktuellen Realisierung auf Basis des RTOS-UH können die Steuer- und Messvorschriften gelegentlich Fehler verursachen, welche das gesamte System negativ beeinflussen können.

Im Rahmen dieser Systemumstrukturierung müssen große Teile des Systems angepasst werden. Infolgedessen hat das Unternehmen beschlossen, auch die aktuell genutzte Programmiersprache PEARL90 durch C/C++ zu ersetzen, weil diese einen umfangreicheren Pool an Bibliotheken bietet. Die Firma ESD arbeitet derzeit an der Portierung des vorhandenen Quellcodes um ein komplettes Neuschreiben zu vermeiden. Derzeit werden die Steuer- und Messvorschriften noch in PEARL90 entwickelt. Diese sollen in Zukunft grafisch modelliert werden. Durch diese Modellierung soll das Erstellen, Bearbeiten und Pflegen der SMV vereinfacht werden.[79]

3.5 Problemanalyse

Um die Ursachen des vorhandenen Kommunikationsproblems zwischen Prüftechnik und Datenbank untersuchen zu können, wird im Folgenden das bereits in Kapitel 2.1.5 eingeführte Qualitätswerkzeug „Ishikawa-Diagramm“ eingesetzt. Dieses zu nutzen bietet sich besonders an, da so das Problem übersichtlich analysiert werden kann.

[79] http://ess.cs.tu-dortmund.de/Teaching/Theses/2011/BA_Knobe_2011.pdf

Folgende Ursachen können nach Absprache mit den projektinvolvierten Kollegen gefunden werden:

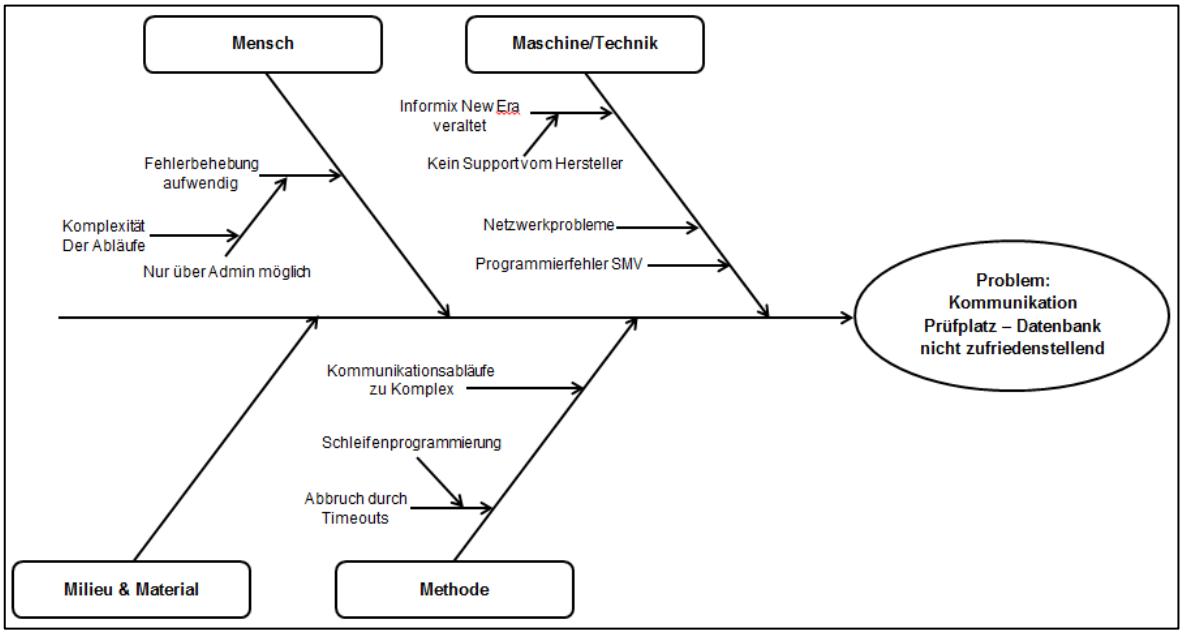


Abbildung 31 Ishikawa Ist-Zustand

Als Problem wird hier, gemäß der Aufgabenstellung, die nicht zufriedenstellende Kommunikation zwischen Prüfplatz und Datenbank genannt. Die Ursachen werden in die 3 Hauptursachen Mensch, Maschine und Methode unterteilt. Da sowohl für den Bereich Milieu als auch Material keine Ursachen gefunden wurden, bleiben diese leer. Die einzelnen Ursachen werden in den folgenden Unterkapiteln näher definiert und analysiert.

3.5.1 Maschine/Technik

Durch die im Kapitel 3.4.1 beschriebene Umstellung von der Dauerfeldversuchsdatenbank (DFVDB) auf das ProLab/LabDB-System muss die Kommunikation generell angepasst werden. Eine einfache Übernahme des Kommunikationsprotokolls ist nicht möglich. Die Softwareumstrukturierung bietet eine hervorragende Möglichkeit, den kompletten Kommunikationsprozess zu überdenken und gegebenenfalls umzustrukturieren.

Des Weiteren kommt es immer öfter zu Problemen innerhalb des Netzwerkes, bedingt durch die komplexen Vorgänge und die Steuer- und Messvorschriften werden nicht immer sauber programmiert, sodass immer wieder Anpassungen vorgenommen werden müssen.

3.5.2 Methode

Auch methodische Problemursachen können in den Kommunikationsabläufen gefunden werden. Um diese weiter zu analysieren und da die Kommunikationsabläufe essenziell für diese Arbeit sind, wird der derzeitige Kommunikationsprozess näher beschrieben.

Die Datenübertragung zwischen den Prüfplätzen und der Datenbank kann im Wesentlichen in vier Hauptprozesse und insgesamt vier Unterprozesse unterteilt werden. Diese sind:

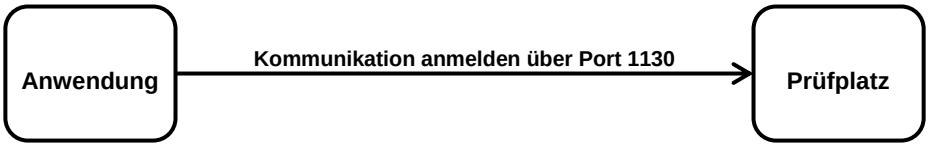
- **Prüfplatz einrichten**
    - Prüfplatz starten
    - Prüfplatz fortsetzen
- **Prüfplatz anhalten**
    - Prüfplatz unterbrechen
    - Prüfplatz anhalten
- **Datensatz senden**
- **Aktuellen Zustand senden**

Um die derzeitige Kommunikation zu verdeutlichen, folgt nun eine Analyse des „Prüfplatz einrichten“-Prozesses. Es ist für diese Arbeit nicht nötig, jeden einzelnen Prozess detailliert zu erläutern. Dieser dient beispielhaft um das Prinzip zu verdeutlichen. Alle anderen Prozesse laufen ähnlich ab und die aus dieser Analyse geschlussfolgerten Probleme lassen sich in allen Prozessen wiederfinden bzw. auf diese übertragen.

Der grundlegende Aufbau der Kommunikation besteht abstrahiert aus folgenden drei Grundelementen:

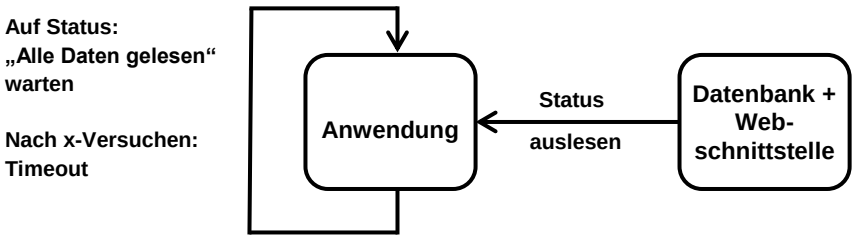
Die Anwendung und der Prüfplatz kommunizieren nicht direkt miteinander, sondern über eine dazwischen liegende Datenbank und einer Webschnittstelle.

Als erster Prozessschritt wird vom Prüfpersonal in der Anwendung (DFVDB) eine Kommunikation zu dem gewünschten Prüfplatz angemeldet. Dazu wird der Prüfplatz über einen definierten Port aufgerufen. Der Prüfplatz ist dann in einem Kommunikationsmodus und wartet auf weitere Informationen. Die weitere Kommunikation zwischen Anwendung und Prüfplatz läuft nun über die Datenbank.

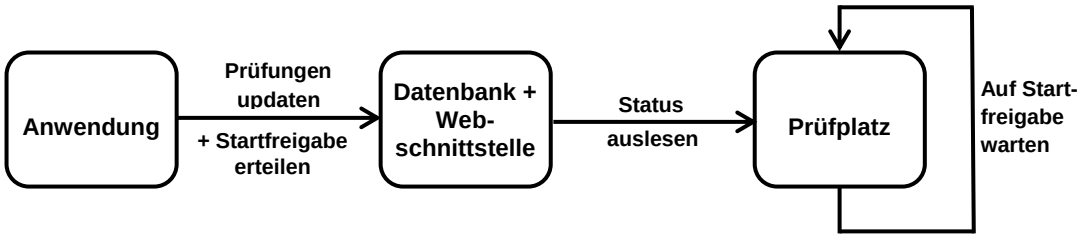


Die Anwendung erzeugt in der Datenbank einen sogenannten Sysjob, welcher von dem Prüfplatz ausgelesen wird. Der Sysjob ist ein Datensatz in einer Datenbanktabelle, in der alle nötigen Daten während der Kommunikation zwischen Prüfplatz und Anwendung gesammelt und ausgetauscht werden. Die JavaServer-Page (JSP) DBAccess4PPR.jsp stellt die Webschnittstelle dar, über die der Prüfplatz auf die Datenbank zugreift. Sie läuft auf einem Tomcat-Webapplicationserver. Diese JSP ist mit einem Link auf eine Website vergleichbar. Der prüfplatzinterne PEARL-Code ruft an einer bestimmten Stelle diese JSP auf und übergibt prüfungsabhängige Parameter. Die JSP überträgt diese Parameter dann an die Datenbank.

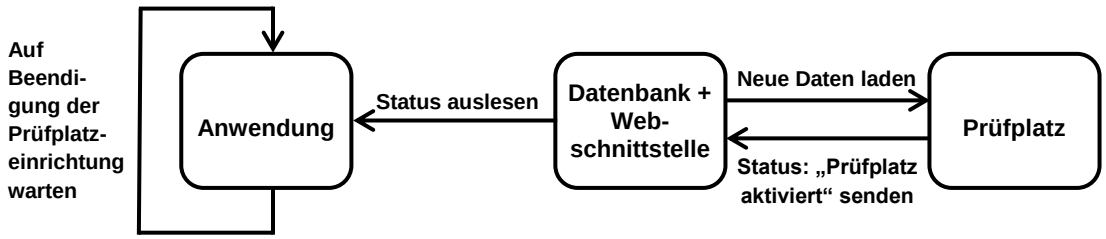
Während der Prüfplatz neben dem Sysjob auch alle nötigen Vorgabedaten über die Wäschebeladung und die Unwucht einliest, wartet die Anwendung auf den Status „Alle Daten gelesen“, welcher der Prüfplatz nach erfolgreichem Einlesen setzt. Die Wäschebeladungstabelle beinhaltet dabei Vorgaben bezüglich der Art von Wäsche, der Anzahl von Waschzyklen und der Waschmitteldosierung, die für die anstehende Prüfung geplant ist. Mit Hilfe der Unwuchttabelle können Informationen zur vorhandenen Unwucht im Prüfobjekt, entstehend durch ungleichmäßige Verteilung der Wäsche in der Trommel, generiert werden. Diese Unwuchtdaten werden mit den gerätinternen Unwuchtwerten verglichen.



Ist der Status „Alle Daten gelesen“ gesetzt, wartet der Prüfplatz auf eine Startfreigabe durch die Anwendung, welche erst nach erfolgreichem Update der Prüfungen erfolgt.



Ist die Freigabe erteilt, aktiviert der Prüfplatz die erforderlichen neuen Daten und wartet auf die Steuer- und Messvorschrift (SMV). Diese Vorschrift beschreibt den Ablauf einer jeden Prüfung. In ihr sind somit alle Informationen gespeichert, wie das Prüfobjekt zu prüfen ist. Nun ist der Prüfplatz aktiviert und dies wird auch in der Tabelle sys_job dokumentiert.



Anschließend wird die eingerichtete Prüfung innerhalb der Anwendung angezeigt und die Prüfung kann beginnen.

Dabei ist zu beachten, dass die Anwendung, während sie auf Antwort des Prüfplatzes wartet, nur eine bestimmte Anzahl an Anfrageversuchen erlaubt. Ist die Anzahl überschritten, führt dies zu einem Timeout und die Kommunikation wird unterbrochen, da es sonst zu einer Überlastung des Netzes kommen könnte. Kritisch dabei ist, dass nicht sofort eine neue Kommunikation aufgebaut werden kann, da der Prüfplatz selbst noch auf die Freigabe wartet. Solange ist sein Aktivierungsport gesperrt und eine erneute Verbindung mit der Anwendung unmöglich. Erhält der Prüfplatz die erwartete Freigabe innerhalb einer definierten Zeit nicht, so fällt auch er in einen Timeout. Erst jetzt kann ein neuer Versuch der Prüfplatzeinrichtung unternommen werden.

Kritisch für das System ist die komplette Kommunikation über die Datenbank und den Webservice zu betrachten, da sie zu Komplex ist und an dieser Stelle oft Fehler auftreten können. Außerdem erwies sich neben der Kommunikation zwischen Datenbank und Prüfplatz auch das Warten auf die SMV als störanfällig.

Im Allgemeinen fällt auf, dass die Kommunikation aufwändig und damit fehleranfällig ist. Tritt an einer der zahlreichen Kommunikationsschritte ein Fehler auf, führt dies zum Abbruch der Kommunikation. Dies kostet dem Bedienpersonal Zeit. Es muss seine Arbeitsleistung in die Behebung der Fehler investieren, was die Produktivität drastisch senkt. Dies ist unbedingt zu vermeiden.

3.5.3 Mensch

Die letzte Ursache für eine nicht zufriedenstellende Kommunikation ist in der aufwendigen Fehlerbehebung zu sehen. Wenn der Prüfplatz in Folge von Kommunikationsfehlern eine Störung anzeigt, ist ein schnelles Neustarten nicht möglich. Erst durch Eingreifen eines Administrators mit den nötigen Zugriffsrechten und Fachwissen, kann dieser initialisiert werden. Dies ist nötig, da wie in Kapitel 3.5.2 schon beschrieben, die technischen Abläufe zu umfangreich sind.

Durch ein Vereinfachen der Kommunikation könnte eventuell der Schritt über den Administrator umgangen werden. Im Idealfall würden gar keine Störungen mehr entstehen bzw. würde der Prozess sich selbst erhalten und ein manuelles Eingreifen würde generell entfallen.

4 Problemlösung

4.1 Möglichkeiten der technischen Umsetzung

Für die Lösung des analysierten Problems muss sich zunächst für eine Programmiersprache entschieden werden. Insgesamt werden drei Varianten genauer untersucht, da diese, nach einer Vorauswahl, die höchste Erfolgswahrscheinlichkeit bieten. Diese Möglichkeiten sind:

- Java
- C++
- JavaServer Pages (JSP)

4.1.1 Variante 1: Java

Innerhalb der Abteilung Konstruktion und Entwicklung wurde zu Anfang eine Java-Lösung gewünscht, denn Java bietet den Vorteil der Plattformunabhängigkeit. Dies bedeutet, dass die Programme auf allen Endgeräten in gleicher Weise ausgeführt werden können, also unabhängig von dem verwendeten Betriebssystem oder der Hardware. Dies ist möglich, da der Programmcode in einen Bytecode umgewandelt und dann über eine installierte virtuelle Maschine (VM) ausgeführt wird. Somit wäre in diesem Projekt eine direkte Verbindung zwischen ProLab und dem Prüfplätzen möglich. Ein zwischengeschalteter Server, wie es derzeit der Fall ist, wäre nicht mehr vonnöten. Die Struktur würde vereinfacht werden, da kein drittes System im Prozess involviert ist. Abbildung 32 zeigt den grundlegenden Aufbau bildlich:

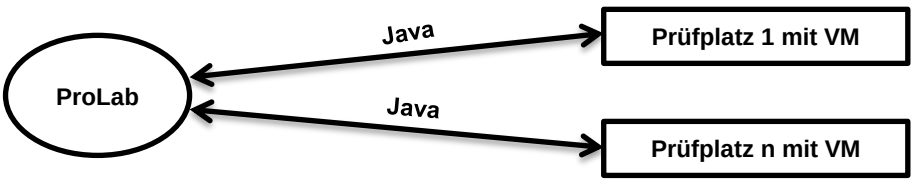


Abbildung 32 Variante 1: Direktverbindung mit Java

Jedoch wurde bisher keine Möglichkeit gefunden eine solche VM auf den Prüfplätzen unter dem neuen Betriebssystem QNX zu installieren, was dazu führt, dass auch kein Java-Programmcode ausgeführt werden kann. Es muss zunächst entweder eine VM gefunden oder andere Alternativen in Erwägung gezogen werden.

Nach Ausgiebiger Recherche konnten insgesamt folgende drei Softwarelösungen ausfindig gemacht werden, welche eine solche VM für QNX anbieten:

- J9 von IBM[80]
- Aonix PERC von Atego[81]
- Jamaica VM von Aicas[82]

Dabei sollte allerdings bedacht werden, dass durch die wenigen Anbieter eine hohe Abhängigkeit an diese vorhanden ist. Die Anbieter können durch das geringe Angebot auf dem Markt hohe Preise für ihr Produkt verlangen. Eine erster Kostenvoranschlag bei Aicas ergab, dass bei 250 Lizenzen Gesamtkosten von 32.500 € (Projektlizenz 15.000 € + 70 € pro Lizenz) entstehen würden. Es ist außerdem als kritisch anzusehen, ob der begrenzte Arbeitsspeicher in Höhe von 64Mb ausreichend wäre.

Da eine Bindung an einen externen Entwickler Abhängigkeiten schafft und die Investitionssumme zu hoch ist, wird diese Programmiersprache keinen Einsatz finden.

[80] http://wiki.eclipse.org/index.php/J9

[81] http://www.atego.com/products/aonix-perc/

[82] http://www.aicas.com/

4.1.2 Variante 2: C++

In der zweiten Variante werden die Prüfplätze ebenfalls direkt an die Anwendungssoftware ProLab angeschlossen. Als Softwaretechnologie soll nun die Programmiersprache C++ Anwendung finden:

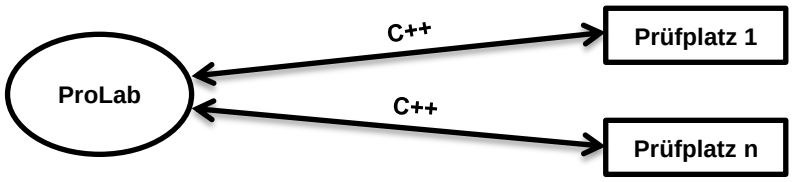


Abbildung 33 Variante 2: Direktverbindung mit C++

Allerdings bietet C++ nicht die gleiche Plattformunabhängigkeit wie Java/JSP und es muss daher genau auf die gewünschten Ansprüche programmiert werden. Da als Datenbank sowohl Informix als auch Microsoft SQL Server nutzbar sein soll, muss hierfür eine geeignete Schnittstelle gefunden werden, denn jede dieser Datenbanken hat seine eigene interne Sprache. Der Datenaustausch wird mittels SQL stattfinden, jedoch muss für das Ansprechen erst ein Mechanismus unter QNX gefunden werden.

Unter dem Betriebssystem Windows oder Linux wäre eine Verbindung über eine sogenannte Open Database Connectivity (ODBC) die einfachste Lösung. ODBC ist eine universelle, von Microsoft entwickelte, Datenbankschnittstelle. In ODBC-basierenden Programmen können problemlos die verwendeten Datenbanken ausgetauscht werden, ohne dass der Programmcode geändert werden muss. Dieser variable Einsatz ist bei den Standardtreibern für den Datenbankzugriff nicht immer möglich. Alle gängigen Datenbanken unterstützen ODBC, allerdings ist es weniger als kaufbares Produkt, sondern vielmehr als ein Standard anzusehen. Die Datenbankentwickler bieten im Normalfall solch ein ODBC-Treiber für die gängigen Be-

triebssysteme kostenlos an.[83] Jedoch ist für QNX in Verbindung mit der Informix-Datenbank kein Treiber zu finden.

Eine andere Möglichkeit könnte das Opensource-Projekt „FreeTDS“ darstellen, mit dem eine Verbindung zu Sybase ASE und Microsoft SQL Server mittels C++ hergestellt werden kann. Sollte dies auch für QNX portierbar sein, würde nach wie vor Zugriff auf die Informix-Datenbank fehlen. Außerdem stagniert das Projekt seit ca. 10 Jahren. Somit ist dies auch keine geeignete Lösung.

Die letzte Möglichkeit könnte eine ODBC-ODBC-Bridge von der Firma Easysoft sein.[84] Dabei handelt es sich um eine Server-Client-Lösung, welche ODBC für viele verschiedene Client-Plattformen ermöglicht. Laut Hersteller wäre es damit z.B. möglich, eine ODBC-Applikation auf einem Linux-Betriebssystem auszuführen, welche einen Microsoft-ODBC-Treiber benötigt, der nur auf Windows verfügbar ist. Als möglicher Client wird unter anderem auch QNX unterstützt. Allerdings erscheint diese Lösung überzogen und aufwendig. Es fehlt jegliche Erfahrung mit solch einem System und es würden Lizenzgebühren anfallen. Dabei wäre auch nicht garantiert, ob das System wirklich lauffähig ist. Das Risiko und die Kosten sind also auch hier zu hoch.

Da keine zufriedenstellende, zuverlässige, sowie kostengünstige Methode für eine direkte Verbindung gefunden wurde, wird auch diese Variante wahrscheinlich keine Lösung darstellen.

4.1.3 Variante 3: JavaServer Pages

Die dritte Möglichkeit für einen Kommunikationsaufbau könnte darin bestehen, einen Server zwischen der Anwendung ProLab und den Prüfplätzen zu schalten.

---

[83] Krause, Jörg; PHP 5 - Grundlagen und Profiwissen: Webserver-Programmierung unter Windows und Linux, 2005, Carl Hanser Verlag GmbH & CO. KG, S.592 ff.

[84] http://www.easysoft.com/products/data_access/odbc_odbc_bridge/index.html

Dieser kann entweder als Hyper-V in der bestehenden Server-Struktur oder in Form eines externen Rechners eingerichtet werden.

Hyper-V ist ein Produkt der Firma Microsoft, welche bei Miele Hausstandard ist. Bei diesem Produkt handelt sich um eine Server-Virtualisierungssoftware, welche es ermöglicht, auf einem hardwaremäßig vorhandenen realen Server mehrere virtuelle Server zu verwalten. Der große Vorteil ist, dass so die Serverauslastung optimiert werden kann. Es ist somit nicht nötig, für kleinere Aufgaben jeweils einen eigenen Server aufzusetzen. Somit sinken die Kosten für Hardware, Instandhaltung und Strom. Die Standardversion von MS Hyper-V Server 2008 R2 ist dabei kostenlos.[85] Nachteil dieser Lösung ist, dass eine Ausfallsicherheit gewährleistet werden muss. Denn sollte die Hard- oder Software dieses Servers ausfallen, würde die Verfügbarkeit von allen darauf installierten virtuellen Maschinen nicht gewährleistet sein. Dies kann zu einem Totalausfall der kompletten Daten-Infrastruktur führen. Daher sollte immer ein Ersatzserver vorhanden sein, welche im Notfall sofort die Aufgaben übernehmen kann.

Als Alternative kann eine herkömmliche Recheneinheit als eigenständiger Server eingesetzt werden. Sollte sich für eine Umsetzung mittels JavaServer Pages entschieden werden, würde zunächst auf solch einem Testserver die Funktionalität überprüft werden. Zu einem späteren Zeitpunkt kann eine Implementierung in die Hyper-V umgesetzt werden.

Auf diesem nun zur Verfügung gestellten Server wird dann die Open Source Software Apache Tomcat[86] installiert. Dieser übersetzt die JavaServer Pages in sogenannte Servlets und führt diese aus. Somit ist keine virtuelle Maschine (VM) auf dem Prüfplatz nötig, da das Übersetzen von dem Tomcat-Server übernommen wird. Tomcat ist der am weitesten verbreitete Application Server und ist derzeit bei Miele im Einsatz.[87]

---

[85] http://www.microsoft.com/de-de/server/hyper-v-server/default.aspx

[86] http://tomcat.apache.org/

[87] http://news.netcraft.com/archives/2003/04/10/java_servlet_engines.html

JSPs sind HTML-Dokumente mit eingebettetem Java-Code. Dies macht die Webinhalte sehr dynamisch. Getrennt werden die Java-Anweisungen vom HTML-Code durch definierte Markierungen (Tags). Der Applikationsserver (Tomcat) übersetzt die JSP in ein Java-Servlet und führt diesen anschließend aus.[88] Die Plattformunabhängigkeit von Java kommt hier besonders zum Tragen, da das System aus einer Informix-Datenbank und dem Betriebssystem QNX Neutrino besteht. Durch die Java-Technologie ist ein problemloses Kommunizieren möglich.

Die folgende Abbildung verdeutlicht den Aufbau:

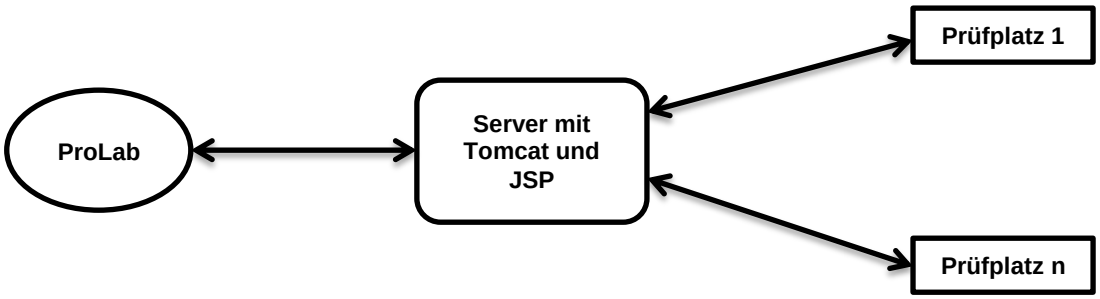


Abbildung 34 Variante 3: Tomcat + JavaServer Pages

Wenn der Prüfplatz nun eine Kommunikation mit der Datenbank bzw. mit ProLab aufbauen möchte, ruft dieser den einprogrammierten Link auf und übergibt die prüfungsspezifischen Parameter. Die JSP regelt dann den kompletten Datentransfer. Diese würde also, angesprochen durch den Prüfplatz, den kompletten Datenaustausch regeln. Der Prüfplatz würde nach dem Aufrufen des Links die Daten als Input erhalten und die interne Programmierung könnte fortfahren.

Der Vorteil dieser Alternative ist, dass eine Programmierung in Form einer JSP recht einfach gehalten ist, da es sich nur um HTML- und Java-Code handelt. Diese sind beide, im Vergleich zu anderen Programmiersprachen, unkompliziert und eine zukünftige Pflege wird vereinfacht. Auch ist die Anwendung ProLab ebenfalls mit JavaServer Pages geschrieben. Durch die Verwendung derselben Programmiersprache kann die Kommunikation direkt in die Anwendung implementiert wer-

[88] Balzert, Helmut; JSP für Einsteiger, W3l.-Verlag, 2003, S.4

den und eine optimale Zusammenarbeit zwischen der Anwendung und der Datenübertragung zum Prüfplatz wird gewährleistet. Außerdem hat das Unternehmen Miele bereits Erfahrung mit Tomcat-Servern, was die Einführung erleichtern würde.

Die derzeitige Kommunikation funktioniert auf ähnlichem Wege. Die Technik ist dabei identisch. Es wird ebenfalls Tomcat als Applicationserver verwendet. Der große Unterschied wird dann vor allem in einem veränderten Prozess zu sehen sein.

Ein Nachteil ist, dass dadurch nach wie vor ein drittes System im Prozess eigegliedert ist, welches eine weitere mögliche Fehlerquelle darstellt. Denn jede zusätzliche Hardware steigert die Ausfallwahrscheinlichkeit des kompletten Prozesses.

Nach Abwägung der drei Möglichkeiten wird sich für eine Lösung in Form einer JSP entschieden, da sie keine weiteren Kosten verursacht, das Unternehmen mit der vorhandenen Technik bereits vertraut ist und die Kommunikation direkt in ProLab integriert werden kann.

4.2 Prozessentwicklung

Im nächsten Schritt muss nun der neue Prozess entwickelt werden. Durch eine gründliche Planung wird eine Struktur erarbeitet, auf deren Basis später die Programmierung vorgenommen werden kann.

Um die Komplexität des Prozesses reduzieren zu können, ist es sinnvoll, aus dem Push-Prozess einen Pull-Prozess zu machen. Derzeit steuert der Anwender die Prüfung von einem PC aus und weißt den Prüfplätzen innerhalb der Anwendung eine Prüfung zu. Die Anwendung sendet dann die gewünschten Daten an den Prüfplätz.

Abbildung 35 Push-Prozess

Der neue Prozess wird dabei den entgegengesetzten Weg gehen. Der Anwender wird die komplette Steuerung, statt am PC, direkt vor Ort am Prüfplatz übernehmen. Nach Eingabe am Bedienelement ruft der Prüfplatz die JSP mit den gewünschten Parametern auf und diese kommuniziert dann direkt mit der Anwendung bzw. der Datenbank.

Abbildung 36 Pull-Prozess

In diesem Prozess ist die Anwendung nun passiv und dient als Anlaufstelle für die JSP. Somit ist der Prüfplatz das ausführende Organ im Prozess und bestimmt selbst, wann er Daten senden und empfangen möchte. Die Komplexität wird deutlich verringert, da keine Schleifen innerhalb der Prüfplatzprogrammierung mehr nötig sein werden.

4.2.1 Modularer Aufbau

Die nächste Überlegung beinhaltet den Funktionsumfang der JSP. Also die Fragestellung, was diese genau leisten muss. Dazu muss zuerst analysiert werden, welche Kernprozesse ausgeführt werden sollen.

Im ersten Schritt ist es nötig, dass ein Prüfplatz mit der Datenbank abgleichen kann, ob für diesen eine Prüfung freigegeben wurde. Somit muss die JSP die Möglichkeit besitzen, mit der Prüfplatznummer nachzusehen, ob eine Prüfung vorhanden ist und die Prüfungsnummer dem Prüfplatz mitteilen (siehe Kapitel 4.2.2).

Ist eine Prüfung freigegeben und dem Prüfplatz übertragen worden, wird dieser eingerichtet. Dafür benötigt die Prüfstation Prüfdaten (Zyklenzahl etc.), Wäschebeladungs- und Unwuchttabellen, sowie Steuer-und Messvorschriften. Diese Daten werden ebenfalls mittels der JSP übertragen. Der Prüfplatz übergibt die zuvor

ausgelesene Prüfungsnummer an die JSP und diese sucht die dazu passenden Daten aus der Datenbank und übergibt sie an den Prüfplatz (siehe Kapitel 4.2.4).

Während der Prüfung fallen Prüfdaten an. Dies sind sowohl Messdaten, als auch ein Fortschritt in Form vom aktuell laufenden Waschzyklus. Diese Daten müssen regelmäßig an die Datenbank gesendet werden. Auch diese Funktionalität muss die JSP beinhalten. Dabei übergibt der Prüfplatz mittels Parameterübergabe die Werte an die JSP und diese ordnet sie der zugehörigen Prüfung der Datenbank zu (siehe Kapitel 4.2.6).

Nun kann es passieren, dass während der Prüfung Änderungen an den Prüfdaten vorgenommen werden müssen. Da die Kommunikation vom Prüfplatz aus aktiviert wird, muss dieser in regelmäßigen Abständen (z.B. am Ende eines jeden Zyklus-es) abfragen, ob sich die zugehörigen Daten innerhalb der Datenbank geändert haben. Wenn das der Fall ist, werden diese Daten geladen (siehe Kapitel 4.2.5) und wenn nicht, wird wie gewohnt weiter geprüft. Dabei dürfen nur die Soll-Werte (Waschprogramm etc.) aktualisiert werden, nicht aber die Ist-Werte (Zyklenzahl, Anzahl Umdrehungen). Der Update-Prozess muss also ein anderer sein, als der Einrichten-Prozess.

Wenn eine Prüfung vom Prüfplatz aktiviert wurde, muss auch dies mit einem JSP-Aufruf der Datenbank mitgeteilt werden, damit diese die Prüfung auf „Aktiv“ setzen kann. Die letzte Funktionalität der JSP übernimmt diese Aufgabe (siehe Kapitel 4.2.3).

Somit ergeben sich für die zu programmierende JSP folgende sechs Hauptfunktionalitäten:

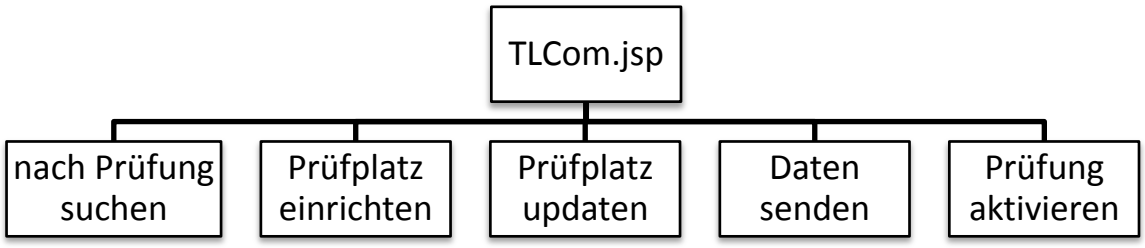


Abbildung 37 Funktionsumfang der neuen JSP

Diese fünf Funktionen werden mit Hilfe eines statischen Include-Befehls in eine JSP integriert und mittels eines Parameters vom Prüfplatz ausgewählt. Dafür macht sich die JSP die URL-Übergabe zu Nutze. Dafür wird hinter die JavaServer Page ein Fragezeichen sowie der Parametername, gefolgt von einem Gleichheitszeichen und dem Parameterwert geschrieben.

Dieser Aufbau sieht wie folgt aus:

File.jsp?parametername=parameter

Der erste Parameter kann beispielsweise eine Funktionsauswahl darstellen. Wenn dieser Parameter einen bestimmten Wert annimmt, wird anschließend der Programmcode an einer definierten Stelle, also im gewünschten Modul, fortgesetzt.

Als Ergebnis erhält man eine einzige JSP, welche in der Lage ist, alle Kommunikationsprozesse zwischen Prüfplatz und Datenbank zu übernehmen. Der Prüfplatz ruft nur diese eine JSP mit den nötigen Parametern auf und erhält bzw. sendet daraufhin die Daten.

Durch den modularen Aufbau gibt es keine unnötigen Wiederholungen innerhalb des Programmcodes. Getreu der Programmierdevise DRY (Don't Repeat Yourself) wird ein Modul nur einmal geschrieben und dann aufgerufen, wenn es gebraucht wird. Würde man für jedes Modul eine eigene JSP schreiben, welche im richtigen Fall direkt aufgerufen wird, wären Wiederholungen unvermeidlich. Sollte zu einem späteren Zeitpunkt eine Änderung nötig sein, z.B. indem man eine Autorisierung des Prüfplatzes mittels MAC-Adresse zu Beginn hinzufügen möchte, muss diese in jede einzelne JSP geschrieben werden. Bei dem modularen Aufbau ist eine Änderung nur einmal nötig.

4.2.2 Modul 1: „Search Test“

Dieses Modul gibt dem Prüfplatz die Nummer einer freigegebenen Prüfung zurück. Dafür werden zwei Parameter benötigt. Einerseits die Auswahl des Moduls und andererseits die Übermittlung der Prüfplatzidentifikation anhand des Client-

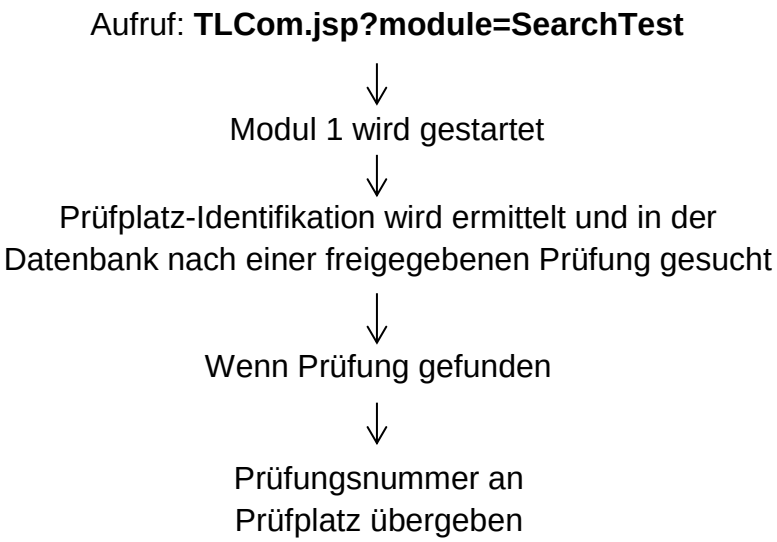
Namens. Dieser Client-Name wird direkt über eine Funktion ermittelt und braucht nicht über eine Parametereingabe übertragen werden.

Mit der Prüfplatzidentifikation wird in der Datenbank nachgeschaut, ob eine Prüfung vorliegt. Dabei kann es sein, dass für einen Prüfplatz mehrere Prüfungen vorliegen. Diese werden alle übertragen und der Prüfplatz kann dann auswählen (z.B. über eine Auswahl am Bedienelement), welche Prüfung ausgewählt werden soll.

Die URL für dieses Modul kann beispielsweise so aussehen:

TLCom.jsp?module=SearchTest

Bildlich dargestellt ergibt sich folgender Programmablauf:



4.2.3 Modul 2: „Activate Test“

Im zweiten Modul sendet der Prüfplatz einen Aktiv-Status in der Datenbank, sobald eine Prüfung gestartet wurde. Dazu muss lediglich die Prüfungsnummer (testno) und die genaue Startzeit im Format „YYYY-mm-dd HH:MM:SS“ übertragen werden.

Der Aufruf sieht dann beispielhaft wie folgt aus:

TLCom.jsp?module=ActivateTest&testno=PNR-P2012133J&activisionTimeStamp=2012-10-17%2015:35:30

Da das Leerzeichen zwischen Datum und Zeit ein Sonderzeichen darstellt, wird dieses laut ISO/IEC 8859-1[89] mit dem Code “%20” ersetzt.

4.2.4 Modul 3: „Setup Testlocation“

Damit ein Prüfplatz seine Arbeit verrichten kann, benötigt dieser eine Vielzahl von Soll- und Ist-Daten. Zum einen werden die Prüfdaten benötigt, also Informationen über den zu prüfenden Gegenstand und den zugehörigen Prüfbedingungen. Des Weiteren werden die Wäschebeladungsdaten und Unwuchtwerte gebraucht und zu Letzt die Steuer- und Messvorschrift (SMV). Erst wenn all diese Daten vollständig übertragen sind, kann mit der Prüfung begonnen werden.

Da diese Daten wiederrum auf der Datenbank liegen, setzt hier die JSP ein. Das dritte Modul regelt die komplette Datenübergabe für das Einrichten des Prüfplatzes. Dabei werden als Parameter das Modul „module“ und die ausgewählte Prüfung „testno“ benötigt. Der Modulaufruf könnte hier beispielhaft folgendermaßen aussehen:

TLCom.jsp?module=SetupTL&testno=PNR-P2012113J

Wurde das Modul korrekt aufgerufen, sucht die JSP nacheinander die nötigen Daten in der Datenbank. Dabei werden die Datensätze als Wertepaar, also Name = Wert, an den Prüfplatz ausgegeben. Sind alle Übergabewerte übertragen, arbeitet der Prüfplatz wie gewohnt weiter.

[89] http://www.uni-protokolle.de/Lexikon/ISO_8859-15.html

Das nachfolgende Ablaufdiagramm visualisiert den Prozess recht deutlich:

Aufruf: **TLCom.jsp?module=SetupTL&testno=PNR-P2012113J**

↓

Modul 3 wird gestartet

↓

Mit der Prüfungsnummer PNR-P2012113J
in der Datenbank nach Prüfdaten, Wäschebeladung,
Unwuchttabelle und SMV
suchen

↓

Daten in Wertepaare formatieren

↓

Datensammlung an Prüfplatz ausgeben

4.2.5 Modul 4: „Update Testlocation“

Die Idee hinter diesem Modul ist, dass es eine Möglichkeit geben muss, während der Prüfungen auch Änderungen im Ablauf oder der Messung vorzunehmen. Da es sich im Dauerversuch um sehr lange Messzeiträume handelt, wäre ein Abbrechen und Neustarten der Prüfung mit neuen Parametern nicht praktikabel, denn dadurch würden bisherigen Messergebnisse gelöscht. Sinnvoller ist es, nach jedem Zyklus zu überprüfen, ob es neuere Vorgabedaten gibt und diese dann gegebenenfalls zu laden und die Prüfung fortzusetzen.

Dabei muss an dieser Stelle erstmals eine Unterscheidung zwischen Ist- und Sollwerte vorgenommen werden. Sollwerte machen hierbei den Großteil aus. In ihnen sind die Informationen enthalten, was mit der Prüfung erreicht werden soll. Dazu zählen Informationen, welches Waschprogramm zu welchem Zeitpunkt gestartet wird, welche Werte gemessen werden etc.

Im Gegensatz dazu beschreiben die Ist-Werte, was der Prüfplatz in der kompletten Prüfung bereits geleistet hat. Dies sind zum Beispiel die Anzahl der gelaufenen Zyklen und Umdrehungen.

Die Ist-Werte dürfen bei einem Update der neuen Werte auf keinem Fall neu geladen werden, da sie dann auf null zurückgesetzt würden. Nur die Soll-Werte dürfen erneuert werden. Daher ist es notwendig, ein eigenes Modul bereit zu stellen und nicht das Modul „SetupTL“ (siehe Kapitel 4.2.4) mit den neuen Werten zu starten.

Die Unterscheidung, welche Daten als Ist- und Soll-Werte deklariert sind, wird von der Datenbank durchgeführt.

Der Ablauf ähnelt dem aus Kapitel 4.2.4. sehr, weshalb auf eine weitere Beschreibung des Prozesses verzichtet wird. Aufgerufen wird das Modul exemplarisch mit folgender URL:

TLCom.jsp?module=UpdateTL&testno=PNR-P2012113J

Nach dem Aufruf des Moduls werden nur die Sollwerte an den Prüfplatz als Wertepaare übergeben.

4.2.6 Modul 5: „Send Data“

Nachdem sich die bisherigen Module mit dem Lesen innerhalb der Datenbank beschäftigt haben, muss es auch eine Möglichkeit für den Prüfplatz geben, seine Prüfdaten der Datenbank zu übermitteln.

Diese Aufgabe übernimmt das Modul 5 „Send Data“. Eine größere Schwierigkeit stellt dabei das Datenvolumen dar. JavaServer Pages bieten zwei Wege für den Input von Werten. Die einfachste Variante ist die GET-Methode, also wie bisher bei der Modulauswahl als Anhang in der URL (*TLCom*.jsp?Parameter1=Paramter&Parameter2=Parameter etc.). Die URL kann bei vielen Datenpaaren sehr lang werden. Jedoch kann, je nach Browser, nur eine gewisse Anzahl an Zeichen übergeben werden. So liegt die Begrenzung beim Internet Explorer beispielsweise bei 2.083 Zeichen.[90] Auch der Webserver kann nur eine gewisse Zeichenlänge verarbeiten, diese liegt aber im Normalfall über der

[90] http://support.microsoft.com/kb/208427/de

vom Browser.[91] Da einige Zeichen für den Pfad beansprucht werden, sollte von einer verfügbaren Länge von ca. 2.000 Zeichen ausgegangen werden. Die Prüfdaten überschreiten diese bei weitem.

Eine Lösung könnte die POST-Methode sein, bei der die Werte im Body-Teil der Anfrage mitgesendet werden. Angewandt wird diese Methode meist bei umfangreicheren Formularen. Allerdings muss der Browser auch in der Lage sein, ein solches Formular zu erzeugen, da nur so eine POST-Methode genutzt werden kann. Der prüfplatzinterne Browser kann dies leider nicht.

Daher muss weiterhin mit der GET-Methode gearbeitet werden. Um eine unbegrenzte Zeichenlänge übertragen zu können, werden die Daten in mehreren Aufrufen gesendet. Dies wird mit einer Dreiteilung des Moduls möglich.

Zu Beginn wird ein Start-Statement aufgerufen (TLCom.jsp?module=DataStreamStart). Dies dient dazu, den Beginn der Übertragung vorzubereiten. Außerdem wird hier die sogenannte Session-ID ausgelesen und an den Prüfplatz übergeben. Die Session-ID dient dazu, die aktuelle Sitzung (Session) beizubehalten und somit mehrere Anfragen (Requests) innerhalb einer Sitzung durchzuführen. Normalerweise wird die Session-ID per Cookie übertragen.[92] Da der Prüfplatz keine Cookies unterstützt, muss die ID über den Parameteraufruf mitgeteilt werden. Im weiteren Verlauf kann die JSP nun Daten (in diesem Fall die Prüfdaten) in der Session speichern und abrufen.[93] Würde man die Session-ID nicht übermitteln, würde bei jedem weiteren Aufruf eine neue Session gestartet und ein Sammeln von Daten wäre nicht möglich.

Danach werden die Daten mittels des Moduls: „DataStream“ und angehangenen Parametern, unter Angabe der Session-ID, übertragen. Der Aufruf könnte beispielhaft so aussehen:

---

[91] http://www.boutell.com/newfaq/misc/urllength.html

[92] http://www.jsptutorial.org/content/session

[93] Balzert, Helmut, JSP für Einsteiger, S.239

TLCom.jsp;sessionid=811E63EC8FEF491A8F835689743DE9EB?module=DataStream&Prüfdatenname1=Prüfdaten&Prüfdatenname2&Prüfdaten

Dieses Modul wird dann so oft mit unterschiedlichen Parametern aufgerufen, bis alle Prüfdaten in der JSP-Session gespeichert sind.

Sind die Daten vollständig Übertragen, wird dies mit dem Ende-Statement (module=DataStreamEnd) abgeschlossen, woraufhin die Daten in einem Datenpakt an die Datenbank gesendet werden.

Den Ablauf kann man sich wie folgt vorstellen:

Aufruf: **TLCom.jsp?module=DataStreamStart**

↓

Datenempfang wird vorbereitet und Session-ID an Prüfplatz übergeben Input

↓

Aufruf: **TLCom.jsp;sessionid=sessionid?module=DataStream&Daten1[…]**

↓

Daten werden in der Session gespeichert

↓

Aufruf: **TLCom.jsp;sessionid=sessionid?module=DataStream&Daten2[…]**

↓

Daten werden in der Session gespeichert

↓

Aufruf: **TLCom.jsp;sessionid=sessionid?module=DataStreamEnd**

↓

Prüfdaten werden als Wertepaare an Datenbank übergeben

Das Schaubild in Anlage 1 zeigt den Aufbau des gesamten Programmes übersichtlich auf einem Bild zusammengefasst.

4.3 Programmierung

4.3.1 Entwicklungsumgebung NetBeans IDE 6.1

Eine Entwicklungsumgebung dient dazu, effizient und schnell Programme zu erstellen. Es ist zwar möglich, in den meisten Programmiersprachen den Code ohne solch eine IDE (integrated development environment) zu erstellen und z.B. über die Konsole zu kompilieren, jedoch ist dies äußerst unkomfortabel und wird in der Praxis kaum Anwendung finden.

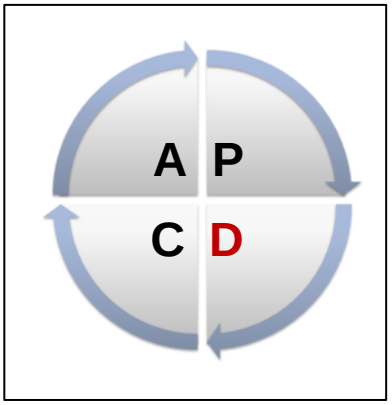


Abbildung 38 Gliederungspunkt Do

In einer IDE ist meist sowohl ein Texteditor, welche die Eingabe von Codezeilen deutlich vereinfacht, als auch Compiler, Linker, Debugger und Quelltextformatierungsfunktionen enthalten.

In der Abteilung KEW der Firma Miele wird vorwiegend mit der NetBeans IDE 6.1 gearbeitet. Da auch ProLab ein mit NetBeans angelegtes Projekt ist, wird die folgende Programmierung ebenfalls mit dieser IDE erstellt, denn eine einheitliche Umgebung erleichtert das gemeinschaftliche Arbeiten an einem Projekt.

Netbeans ist ein von Sun Microsystems gegründetes und 2010 von Oracle übernommenes Open-Source Projekt und befindet sich in einer permanenten Weiterentwickelung. Die Entwicklungsumgebung wurde zwar komplett in Java geschrieben, kann jedoch für viele weitere Programmiersprachen genutzt werden. Netbeans ist kostenlos erhältlich und es gibt keine Nutzungsbeschränkungen.[94]

4.3.2 Einbindung im Projekt und Vorgaben seitens Miele

Die Kommunikation ist Teil der von Miele entwickelten ProLab-Anwendung und wird auch in diese direkt integriert. Ein externes Projekt zu erstellen, welches nur für die Kommunikation zuständig ist, wäre nicht sinnvoll, da die Methoden und Ob-

[94] http://netbeans.org/index_de.html

jekte bereits im Projekt vorhanden sind und so gemeinsam genutzt werden können.

Es ist also zunächst nötig, dass ProLab-Projekt auf dem PC einzurichten und mit allen nötigen Bibliotheken zu verbinden. Dazu werden alle nötigen Zugriffsrechte erteilt und die Ordnerstruktur für die eigenen Dateien erzeugt.

Die Projektstruktur sieht dabei wie folgt aus:

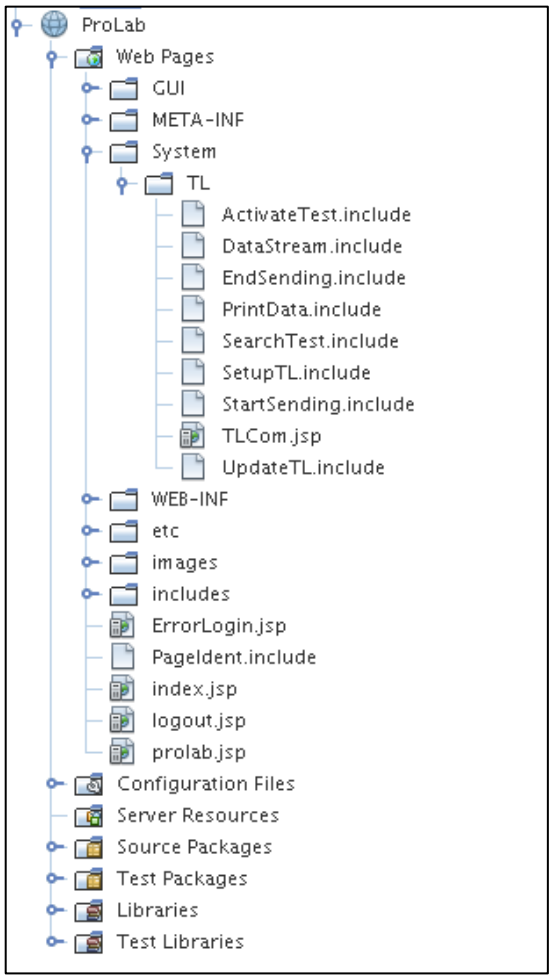


Abbildung 39 Projektstruktur TLCom.jsp

Im Verzeichnis ProLab/Web Pages/System/TL werden alle erstellten Programmdateien gespeichert. Zusätzliche Dateien wie die „index.jsp“ zum Starten von ProLab oder die komplette GUI sind ebenfalls im Projekt enthalten.

Da innerhalb des ProLab-Projektes mehrere Entwickler programmieren, ist es wichtig, sich auf Konventionen für die Gestaltung und den Aufbau des Programmcodes zu einigen. Jeder Programmierer entwickelt in seiner Laufbahn einen eige-

nen Programmierstil. Wenn sich projektfremde Personen nun in den Code einarbeiten möchten, erschwert ein wechselnder Stil die Verständlichkeit. Durch gemeinsame Absprachen wird gewährleistet, dass der Code trotz mehrerer Programmierer, möglichst einheitlich und kompakt wirkt.

Grundlegend gelten natürlich die von Oracle vorgeschlagenen Namenskonventionen, wie z.B. das alle Variablen und Methoden mit einem Kleinbuchstaben und Klassen mit einem Großbuchstaben beginnen sollten.[95] Außerdem sollten alle vergebenen Variablen, Konstanten und Objekte mit sprechenden Namen versehen werden. Das heißt mit Namen, aus denen deren Funktion ersichtlich wird. Auf Variablen wie „i“ oder „k“ sollte daher verzichtet, sondern Namen wie „numberOfLoops“ verwendet werden.

Zusätzliche Vorgaben Seitens Miele sind:

- Jeglicher Text (Variablen, Kommentare, Erläuterungen) muss in englischer Sprache erstellt werden.
- Der Codeaufbau muss sich an einem vorgegebenen Grundskelett orientieren.
- Geschweifte Klammern beginnen immer direkt unter dem zugehörigen Befehl und alle ihm unterstellten Befehle werden mit einem Tabulatorschritt eingerückt. Zum Beispiel:

```
if ( Bedingung)
        {
                Statement
        }
```

- Jede Datei mit Autor-Name, Versionsnummer und Datum, sowie vorgenommene Änderungen im Kopfbereich dokumentieren.

Das Grundskelett hat dabei folgende Elemente, welche auch in exakter Reihenfolge in jeder Code-Datei vorhanden sein soll:

[95] http://www.oracle.com/technetwork/java/codeconventions-135099.html

Tabelle 5 Elemente des Programmcodes

Imports	Alle nötigen Pakete und Klassen werden hier eingefügt
Documentation	Dokumentationsbereich für Programmrelevante Informationen wie z.B. Name, Erstellungsdatum, nötige Parameter, Beschreibung der Funktion usw.
initialize DataHandlingBean + database connection	Initialisieren der “DataHandlingBean” und des „DBConPoolObjTL“
final values for page configuration	Alle Konstanten werden hier aufgelistet
Page Content	Hier beginnt der eigentliche Programmcode; Hauptteil des Programms

4.3.3 Programmaufbau

Auf ein komplettes Erläutern des Programmcodes wird zugunsten der Lesbarkeit der Arbeit im Folgenden verzichtet. Vielmehr werden aufgetretene Probleme erläutert und deren Lösung beschrieben.

Nachdem das Projekt eingerichtet und die JSP an der nötigen Struktur angepasst wurde, kann nun damit begonnen werden, den modularen Aufbau, wie in Kapitel 4.2.1 beschrieben, abzubilden. Dies wird mit der **Include-Funktionalität** erreicht. Ein Include ist ein Verweis auf eine extern ausgelagerte Datei, welche Programmcode enthält. Durch den Einbau dieser Include-Anweisung wird genau an der Stelle im Programmcode der einzufügende Code bereitgestellt. Dies hat einerseits den Vorteil, dass das Programm übersichtlicher wird, andererseits kann man so Wiederholungen vermeiden, da auf eine Include-Datei mehrmals zugegriffen werden kann. Daher ist es sinnvoll, einerseits die Module, andererseits häufig benötigte Funktionalitäten auszulagern.

Dabei unterscheidet man grundsätzlich zwischen zwei Formen von Includes: dynamische (*<jsp:include page="File" flush="true">*) und statische *(<%@ include file="File" %>*). Bei dem statischen Include werden alle Quellcodes zusammengeführt, als wäre es ein Programmcode und dann am Ende zu einem Servlet kompiliert. Im Gegensatz dazu wird der dynamische Include erst zur Laufzeit des Servlets verarbeitet.[96] Dies hat zur Folge, dass das Statische einen direkten Zugriff auf alle Imports und Variablen der Haupt-JSP hat, während das Dynamische alle Importanweisungen selbst beinhalten muss, dafür ist diese selbst voll funktionsfähig und unabhängig vom Hauptprogramm. Für die vorliegende Aufgabe ist ein statisches Include verwendet worden, da eine Eigenständigkeit nicht nötig ist.

Ein weiteres Problem, welches durch die Programmierung gelöst werden muss, sind die **Zugriffsrechte**. Es muss verhindert werden, dass jemand die JSP an einem PC im Webbrowser startet und dann einerseits Daten beliebig auslesen und andererseits auch Daten in der Datenbank schreiben kann. Dies könnte zu großen Schäden führen. Auch würde somit die Möglichkeit für Manipulationen gegeben sein. Um das zu verhindern, wird im ersten Schritt der Client-Name mit der Standardmethode *request.getRemoteHost()* abgefragt. Mit der ProLab-Methode *hasAccess()* wird dann auf der Datenbank abgeglichen, ob dieser Client eine Zugriffsberechtigung besitzt. Zusätzlich greift eine zweite Sicherheitsüberprüfung: die sogenannte ComID.

Die ComID ist eine Art Passwortabfrage zwischen dem Host (ProLab) und dem Client (dem Prüfplatz). Diese ID ist nur zwischen diesen beiden Teilnehmern bekannt und für jeden Client unterschiedlich. Jede größeren Aktion (Daten senden, Daten empfangen etc.) gibt als Return-Statement eine neue ComID zurück. Diese wird von ProLab durch die Befehlszeile:

Double comIDPart = Runtime.getRuntime().hashCode()(Math.random()*10000);*

[96] http://www.jsp-develop.de/tipps/

generiert. Sie setzt sich also aus dem HashCode (eine Integerzahl, welche das Objekt eindeutig identifiziert[97]) der Runtime und einer Zufallszahl zusammen.

Die ComID ist nur für einen Aufruf gültig und wird dann verändert. Dieses neue Passwort muss dem Prüfplatz immer mitgeteilt werden, da er es für die nächste Aktion als Parameter übergeben muss. Jeder URL-Aufruf, der in Kapitel 4.2 beschrieben wurde, muss also mit dem Parameter „ComID=[…]“ erweitert werden. Somit heißt der vollständige Aufruf für das erste Modul Search Test (siehe Kapitel 4.2.2):

TLCom.jsp?ComID=ABC&module=SearchTest.

Sowohl der richtige Client-Name, als auch die ComID sind nötig, um eine Verbindung zu ProLab aufbauen zu können. Ist einer der beiden Werte nicht korrekt, wird der Zugriff verweigert und der weitere Programmcode in der JSP kann nicht ausgeführt werden.

Vor dem Beginn des Hauptteils der Programmierung muss nun noch sichergestellt werden, dass das Programm bei einem Fehler nicht ungewollt beendet wird. Es müssen zu jeder Zeit vorbestimmte Rückgabewerte an den Prüfplatz übermittelt werden. Auch bei einer Fehleingabe darf es nicht einfach zu einem direkten Abbruch kommen, sondern ein kontrolliertes Beenden wäre wünschenswert. Typischerweise wird dies in Java mittels dem sogenannten **„Exception-Handling“** gelöst.

Beim Exception Handling werden innerhalb eines vorher definierten Bereichs Ausnahmen (Exceptions) abgefangen und anschließend behandelt. Je nach Art der Exception kann dann z.B. eine Meldung ausgegeben oder eine spezielle Methode aufgerufen werden.[98] Gekennzeichnet wird der zu prüfende Bereich dabei mit dem Tag *try { }*. Wenn es innerhalb dieses Blockes zu einem Fehler kommt, wird die Exception, welche im darauf folgenden Catch{ } -Block definiert ist, geworfen

[97] Java ist auch eine Insel, Galilieocomputing, http://openbook.galileocomputing.de/javainsel9/javainsel_09_003.htm#mj7fe8f3d9fd296c97017de07898eebccc

[98] Kiwitter, Patrick; Eclipse in der Java Entwicklung; S.205

(*throw*). Wenn ein Fehler auftritt, wird alles innerhalb des Catch-Blockes ausgeführt. Läuft das Programm fehlerfrei durch, wird keine Exception geworfen und der Catch-Block übersprungen.

Ergänzend zu diesen zwei Bereichen kann noch ein dritter, der *finally { }* –Block, eingebaut werden. Der Programmcode in diesem Bereich wird immer ausgeführt, egal ob ein Fehler auftritt oder nicht.

In der JSP werden alle drei Bereiche verwendet. Nach dem Verbindungsaufbau zur Datenbank/ProLab folgt ein try-Block, welcher die komplette Modulauswahl und –verarbeitung umfasst. Wenn ein Fehler auftritt, wird die Exception geworfen, welche eine Ausgabe in Form

„- Exception -
e.getMessage()“

enthält. Der Prüfplatz kann dann entsprechend der Fehlermeldung, welche Variabel den Fehler ausgibt, reagieren und z.B. auf dem Bedienelement die Ursache ausgeben.

Im darauf folgenden Finally-Block wird dann die Verbindung mit *.closeConnection()* geschlossen, unabhängig davon, ob ein Fehler auftrat oder nicht. Somit wird das Programm definitiv sauber beendet.

Nun ist der Programmcode fertig entwickelt und kann im Browser testweise gestartet werden, indem die URLs samt Parameter eingegeben werden. Die Tests haben die gewünschten Resultate geliefert und fehlerfrei eine Verbindung mit der Datenbank hergestellt. Der fertige Quellcode ist in den Anlagen 3-14 zu sehen.

5 Fazit und Ausblick

Nun da die Programmierung abgeschlossen ist, gilt es zu überprüfen, inwieweit die Ergebnisse die vorher herausgearbeiteten Probleme lösen. Dazu wurde das aus Kapitel 3.5 erstellte Ishikawa-Diagramm auf den neuen Prozess abgestimmt und Veränderungen farblich gekennzeichnet.

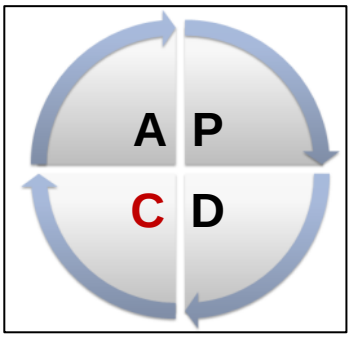


Abbildung 40 Gliederungspunkt Check

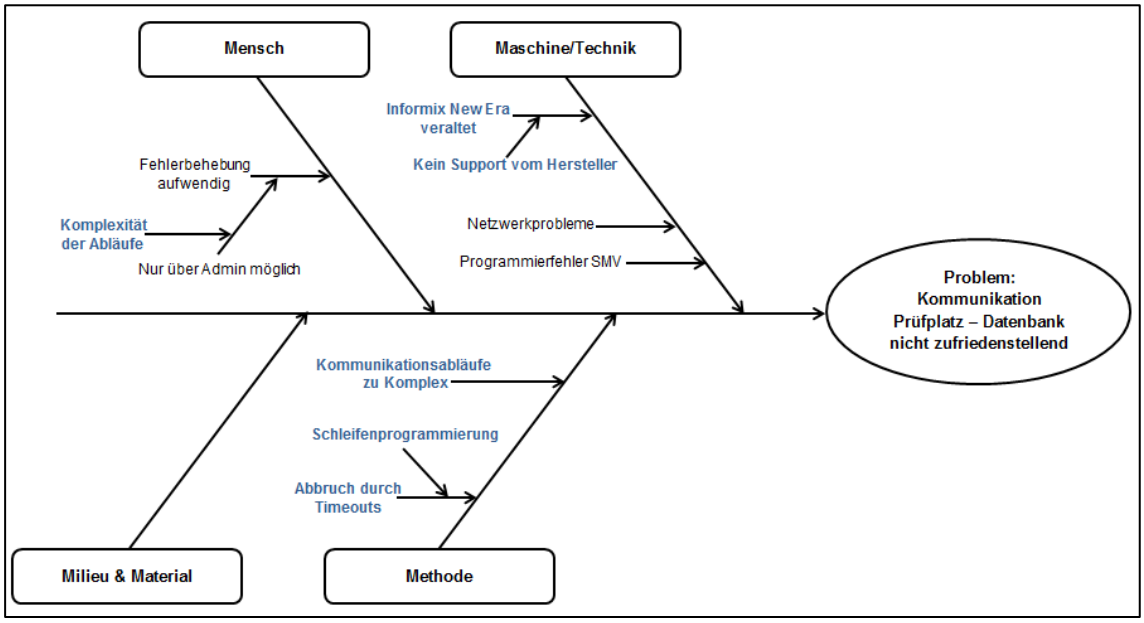


Abbildung 41 Ishikawa Soll-Zustand

Schon auf den ersten Blick wird eine deutliche Verbesserung gegenüber dem alten Prozess deutlich. Die Komplexität wurde deutlich reduziert. Die umfangreichen Kommunikationsschritte zwischen Prüfplatz und Datenbank wurde auf ein minimales Niveau reduziert. Sollten Probleme auftauchen, was erst in einem späteren Praxistest ersichtlich wird, muss jedoch weiterhin der Admin eingreifen. Dieses Problem ließ sich leider nicht lösen.

Durch die konsequente weitere Einführung von ProLab wurde das Problem mit dem mangelnden Support für Informix New Era gelöst. Die Kommunikationsprogrammierung ist nun ein Teil von ProLab. Sie verwenden gemeinsame Methoden und sind gut aufeinander abgestimmt. Da ProLab eine Miele-Eigenentwicklung ist, wird es zukünftig keine Probleme mit dem Support geben.

Innerhalb der Programmierung gibt es keine Schleifen mehr. Das heißt, dass es auch zu keinen Timeouts mehr kommen wird. Der Prüfplatz holt sich selbständig die benötigten Daten und die Datenbank/Datenbankanwendung liefert diese.

Insgesamt wurden viele Probleme gelöst und der Prozess um einiges vereinfacht. Zusammengefasst kann man von einem Erfolg des Projekts sprechen.

Im nächsten Schritt muss nun die Implementierung des neuen Systems vorgenommen werden. Dazu ist es vonnöten, die prüfplatzinterne Programmierung anzupassen, damit die JSP an der korrekten Stelle vom Prüfplatz aufgerufen wird und die Daten verarbeitet werden können. Dies kann allerdings erst nach erfolgreicher Portierung des Pearl90-Codes zu C++ geschehen. Da die Prüfplatzprogrammierung nicht Teil meiner Arbeit ist, wird diese hier auch nicht weiter betrachtet.

Zum besseren Verständnis der Programmierabläufe wurde eine Tabelle erstellt, welche alle Befehle inklusive Input und Output übersichtlich auf einer Seite darstellen. Diese Übersicht ist im Anlage 2 zu finden. Mit deren Hilfe kann sich der Entwickler der Prüfplatzprogrammierung besser in die Kommunikationsprogrammierung einarbeiten und sieht auf einen Blick, welche Inputs er übergeben muss um die gewünschten Outputs zu erhalten.

Literaturquellen

[BaHe2003] Balzert, Helmut: JSP für Einsteiger. Dynamische Websites mit JavaServer Pages erstellen – 1. Aufl. – W3l Verlag, 2003 - 978-3937137018

[BaKö2008] Barnes, David J.; Kölling, Michael: Java Lernen mit BlueJ – 4. Aufl. – Pearson Studium, 2008 - 978-3-86894-001-5

[BeJu2009] Benra, Juliane T.: Software-Entwicklung für Echtzeitsysteme – 1. Aufl. – Springer-Verlag, 2009 - 978-3642015953

[BrJö2007] Brauer, Jörg-Peter.: Qualitätsmanagement von A - Z: Erläuterungen moderner Begriffe des Qualitätsmanagements – 6. Aufl. – Carl Hanser Verlag GmbH & CO. KG, 2007 - 978-3446412736

[CoTh2005] Colsman, Hubertus; Theden, Philipp: Qualitätstechniken: Werkzeuge zur Problemlösung und ständigen Verbesserung – 4. Aufl. – Carl Hanser Verlag GmbH & CO. KG, 2005 - 978-3446400443

[FlDa2005] Flanagan, David: Java in a Nutshell – 5. Aufl. – O'Reilly Media, 2005 - 978-0596007737

[GeFr2011] Geisler, Frank: Datenbanken – Grundlagen und Design – 4.Aufl. – mitp Verlag, 2011 - 978-3826690884

[GeKo2007] Geiger, Walter: Kotte, Willi: Handbuch Qualität: Grundlagen und Elemente des Qualitätsmanagements: Systeme - Perspektiven – 5. Aufl. – Vieweg+Teubner Verlag, 2007 - 978-3834802736

[KaMa2007] Kannengiesser, Matthias: Objektorientierte Programmierung mit PHP 5 – 1.Aufl. – Franzis Verlag, 2007 - 978-3772362965

[KiPa2008] Kiwitter, Patrick: Eclipse in der Java-Entwicklung – 1. Aufl. – Addison-Wesley Verlag, 2008 - 978-3827324900

[KiPr2012] Kirch, Ulla; Prinz, Peter: C++ - Lernen und professionell anwenden – 6. Aufl. – mitp Verlag, 2012 - 978-3826691959

[KrJö2005] Krause, Jörg: PHP 5 - Grundlagen und Profiwissen: Webserver-Programmierung unter Windows und Linux – 2. Aufl. – Carl Hanser Verlag, 2005 - 978-3446403345

[KuTh2007] Kudraß, Thomas: Taschenbuch Datenbanken – 1. Aufl. – Carl Hanser Verlag GmbH & CO. KG, 2007 - 978-3446409446

[LiGe2005] Linß, Gerhard: Qualitätsmanagement für Ingenieure – 2. Aufl. – Carl Hanser Verlag GmbH & CO. KG, 2005 - 978-3446228214

[MaHu2002] Malorny, Christian; Hummel, Thomas: Total Quality Management: Tipps für die Einführung – 3. Aufl. – Hanser Fachbuch, 2002 - 978-3446218635

[PfTi2001] Pfeifer, Tilo: Qualitätsmanagement: Strategien, Methoden, Techniken – 1. Aufl. – Hanser Fachbuch, 2001 - 978-3446215153

[Refa2002] REFA: Ausgewählte Methoden zur prozessorientierten Arbeitsorganisation - REFA-Sonderdruck Methodenteil – REFA-Bestell-Nr. 198213

[RiGr2001] Riccardi, Greg: Datenbanksysteme mit Internet- und Java-Applikationen – 1. Aufl. – Addison-Wesley Verlag, 2001 - 978-3827318756

[SaHe2002] Sauer, Hermann: Relationale Datenbanken: Theorie und Praxis – 1. Aufl. – Addison-Wesley Verlag, 2002 - 978-3827320605

[SaSa2010] Saake, Gunter; Sattler, Kai-Uwe; Heuer, Andreas: Datenbanken – Konzepte und Sprachen – 4. Aufl. – mitp-Verlag, 2010 - 978-3826690570

[ScMa2008] Schneider, Markus: Implementierungskonzepte für Datenbanksysteme – 1. Aufl. – Springer-Verlag, 2008 - 978-3540419624

[ScPe2005] Scholz, Peter: Softwareentwicklung eingebetteter Systeme: Grundlagen, Modellierung, Qualitätssicherung – 1. Aufl. – Springer-Verlag, 2005 - 978-3540234050

[StCh2010] Stroustrup, Bjarne: Einführung in die Programmierung mit C++ – 1.Aufl. – GRIN Verlag, 2010 - 978-3868940053

[StCh2002] Stocker, Christine: Objektorientierte Datenbanksysteme Inhalt, Bedeutung und Beispiel – 1.Aufl. – GRIN Verlag, 2002 - 978-3638138017

[TiMü2003] Tietjen, Thorsten; Müller, Dieter: FMEA Praxis: Das Komplettpaket für Training und Anwendung – 2. Aufl. – Carl Hanser Verlag GmbH & CO. KG, 2003 - 978-3446223226

[TiWo2002] Timischl, Wolfgang: Qualitätssicherung: Statistische Methoden – 3. Aufl. – Fachbuchverlag Leipzig, 2002 - 978-3446220539

[WeJo2003] Weigert, Johann: Der Weg zum leistungsstarken Qualitätsmanagement – 1. Aufl. – Schlütersche Verlag, 2003 - 978-3877066409

[WiJo2008] Wieken, John-Harry: SQL - inkl. Lerntest auf CD: Einstieg für Anspruchsvolle (Master Class) [Taschenbuch] – 1. Aufl. – Schlütersche Verlag, 2008 - 978-3827324856

[ZoHa2011] Zollondz, Hans-Dieter: Grundlagen Qualitätsmanagement: Einführung in Geschichte, Begriffe, Systeme und Konzepte – 1. Aufl. – Oldenbourg Wissenschaftsverlag, 2011 - 978-3486597981

Internetquellen

[AmMA2013] Amies, Mary-Anne: Brainstorming – Wise Up Marketing Solutions: <http://wiseupmarketing.files.wordpress.com/2011/10/brainstormbyyourself.jpg>, verfügbar am 05.01.2013

[AuJü2013] Auer, Jürgen: Sql-und-Xml URL: <http://www.sql-und-xml.de/sql-tutorial/datenbank-grundbegriffe.html>, verfügbar am 05.01.2013

[BlAn2013] Blank, Andreas: iPhone FAQ URL: <http://www.iphonefaq.info/content/12/21/de/wo-wird-das-iphone-produziert.html>, verfügbar am 05.01.2013

[BlWi2013] Blümer, Timo; Marion Winkenbach: Duden.de URL: <http://www.duden.de/rechtschreibung/Qualitaet>, verfügbar am 05.01.2013

[BoTh2013] Boutell, Thomas; Maximale URL-Länge: URL: <http://www.boutell.com/newfaq/misc/urllength.html>, verfügbar am 05.01.2013

[EaLi2013] Easysoft Limited; Easysoft: URL: <http://www.easysoft.com/products/data_access/odbc_odbc_bridge/index.html>, verfügbar am 05.01.2013

[ErKe2013] Erdrich, Ken: Qualitätsmanagement URL: <http://quality.kenline.de/seiten_d/qualitaet_definition.htm>, verfügbar am 05.01.2013

[FaGü2013] Faes, Günter: schiering.org URL: <http://www.faes.de/Basis/Basis-Lexikon/Basis-Lexikon-Histogramm/basis-lexikon-histogramm.html>, verfügbar am 05.01.2013

[FrFl2013] Fröbel, Dirk; Flegel, Jürgen; Uni Potsdam Vor- und Nachteile von Java: URL: <http://ddi.cs.uni-pots-dam.de/HyFISCH/Produzieren/SeminarDidaktik/OOP/java_vor_nachteile.html>, verfügbar am 05.01.2013

[GeW2013] Prof. Dr.-Ing. W. Gerth; Gottfried Wilhelm; Leibnitz Universität Hannover RTOS-UH: URL: <http://www.irt.uni-hannover.de/rtos/>, verfügbar am 05.01.2013

[HaAn2013] Hadler, Andreas; IEP RTOS-UH: URL: <http://www.iep.de/Downloads/deutsch/RTOSstart.pdf>, verfügbar am 05.01.2013

[HoRi2013] Horsman, Richard; atego Aonix Perc: URL: <http://www.atego.com/products/aonix-perc/>, verfügbar am 05.01.2013

[HuJa2013] Dr. Hunt, James; Aicas JamaicaVM: URL: <http://www.aicas.com/>, verfügbar am 05.01.2013

[JaRa2013] Jach, Rainer; Techdivision: URL: <http://www.techdivision.com/blog/agile-projektentwicklung-mit-scrum/>, verfügbar am 05.01.2013

[KnDa2013] Knobe, Daniel; TU Dortmund, Bachelorarbeit: URL: <http://ess.cs.tu-dortmund.de/Teaching/Theses/2011/BA_Knobe_2011.pdf>, verfügbar am 05.01.2013

[KrSc2013] Krome, Hartmut; Schlierenkämper, Rüdiger; Werum Firmenwebsite: URL: <http://www.werum.de; http://www.werum.de/de/mdm/refer/index.jsp>, verfügbar am 05.01.2013

[LeJe2013] Lehmann, Jens: Das Freizeichen; anoxa.de URL: <http://www.anoxa.de/blog2/?p=751>, verfügbar am 05.01.2013

[MiGe2013] Miele-Geschäftsbericht 2011/2012 URL: <http://www.miele-presse.de/media/presse/media/Miele_GB_2011-12_DE_web.pdf>, verfügbar am 05.01.2013

[MiHo2013] Miele-Homepage URL: <http://www.miele.de/de/haushalt/unternehmen/4366.htm>, verfügbar am 05.01.2013

[MlTu2013] Turk, Mladen; Apache Tomcat: URL: <http://tomcat.apache.org/>, verfügbar am 05.01.2013

[NeLi2013] Netcraft Ltd.; Java Servlet Engines: URL: <http://news.netcraft.com/archives/2003/04/10/java_servlet_engines.html>, verfügbar am 05.01.2013

[OrBe2013] Orndorff, Benjamin; Microsoft Server: URL: <http://www.microsoft.com/de-de/server/hyper-v-server/default.aspx>, verfügbar am 05.01.2013

[OrBe2013] Orndorff, Benjamin; Microsoft Maximale URL-Länge: URL: <http://support.microsoft.com/kb/208427/de>, verfügbar am 05.01.2013

[OrCo2013] Oracle Corporation: java.com: URL: <http://www.java.com/de/download/faq/whatis_java.xml>, verfügbar am 05.01.2013

[OrCo2013] Oracle Corporation; Netbeans: URL: <http://netbeans.org/index_de.html>, verfügbar am 05.01.2013

[ReMo2013] Research in Motion Limited; RIM: URL: <http://press.rim.com/newsroom/press/2010/pressrelease-3766.html>, verfügbar am 05.01.2013

[RiWo2013] Rittmeyer, Wolfram; JSP-Tutorial: URL: <http://www.jsptutorial.org/content/session>, verfügbar am 05.01.2013

[RoFr2013] Roeing, Frank: Wikipedia Datenbanksystem URL: <http://upload.wikimedia.org/wikipedia/de/thumb/f/f3/Datenbanksystem.svg/685px-Datenbanksystem.svg.png>, verfügbar am 05.01.2013

[ScWo2013] Schiering, Wolfram: schiering.org URL: <http://www.schiering.org/arhilfen/qualit/qm/1103_03.gif>, verfügbar am 05.01.2013

[SpGu2013] Spickermann, Gunnar: Vergleich der Datenbankmodelle: URL: <http://pi.informatik.uni-sie-gen.de/lehre/2000w/2000w_proseminar/ausarbeitungen/datenbankmodelle.pdf.gz>, verfügbar am 05.01.2013

[StCh2013] Stobitzer, Christian: Datenbank Grundlagen: URL: <http://www.datenbank-grundlagen.de/entity-relationship-modell.html>, verfügbar am 05.01.2013

[ThRa2013] Thome, Rainer: Winfriedschule Fulda Gymnasium URL: <http://www.info-wsf.de/index.php/Datenbankmodelle>, verfügbar am 05.01.2013

[UlCh2013] Ullenboom, Christian; Java ist auch eine Insel: URL: <http://openbook.galileocomputing.de/javainsel9/javainsel_09_003.htm#mj7fe8f3d9fd296c97017de07898eebccc>, verfügbar am 05.01.2013

[UnPr2013] Uni-protokolle; ISO 8859-15: URL: <http://www.uni-protokolle.de/Lexikon/ISO_8859-15.html>, verfügbar am 05.01.2013

[VoKa2013] Prof. Dr. Voigt, Kai-Ingo: Gabler Wirtschaftslexikon URL: <http://wirtschaftslexikon.gabler.de/Definition/qualitaetssicherung.html>, verfügbar am 05.01.2013

[WaBe2013] Wahlen, Bernhard: wahlen-web.info URL: <http://www.wahlen-web.info/beruf_sites/ishikawa.gif>, verfügbar am 05.01.2013

[WaBe2013] Wahlen, Bernhard: wahlen-web.info URL: <http://www.wahlen-web.info/beruf_sites/korrelationsformen.gif>, verfügbar am 05.01.2013

[WaJe2013] Wagener, Jens: Scrum: URL: <http://www.scrum-kompakt.de/grundlagen-des-projektmanagements/vorgehensmodelle-in-der-softwareentwicklung>, verfügbar am 05.01.2013

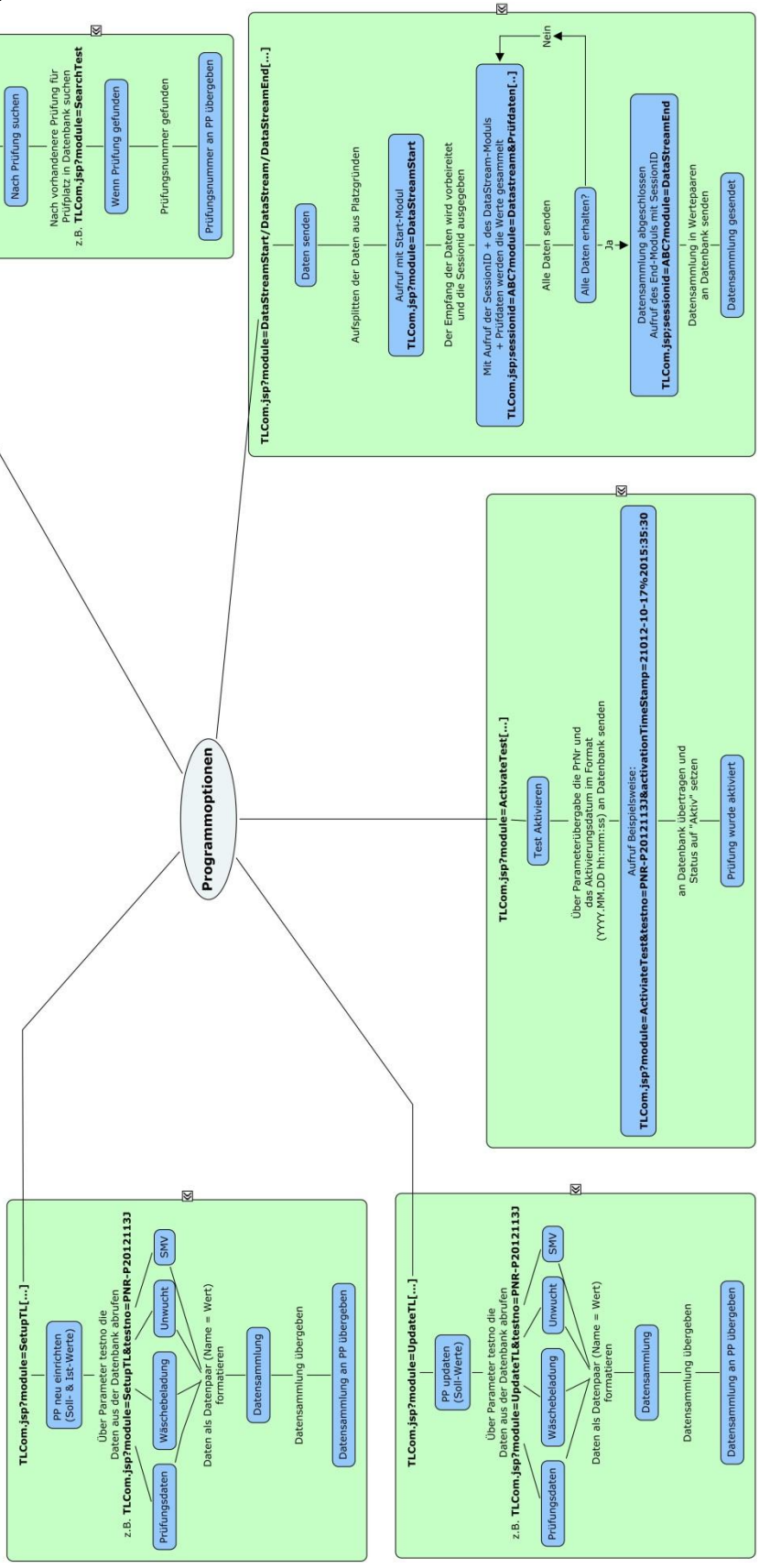
[WeMa2013] Welk, Marc; JSP-Develop: URL: <http://www.jsp-develop.de/tipps/>, verfügbar am 05.01.2013

[WeSc2013] Weggel, Andreas; Schnitte, Andreas; Frenzel, Sandy; QNX: URL: <http://heineck.hof-universi-ty.de/fileadmin/Dozenten/Horst_Heineck/PDF/Echtzeitsysteme/QNX.pdf>, verfügbar am 05.01.2013

[WrDa2013] Wright, Darin; Eclipse J9: URL: <http://wiki.eclipse.org/index.php/J9>, verfügbar am 05.01.2013

Anlagen

Anlage 1, Aufbau JSP



Anlage 2, Übersicht Input-Output

Aufgabe	Aufruf	Eingabeparameter am Beispiel	Ausgabe	Beschreibung
Nach Prüfung suchen	TLCom.jsp?ComID=[…]&module=SearchTest	ComID= ABC123 module= SearchTest	Prüfungsnummern (testno) + Prüfungstitel (testtitle) untereinander aufgelistet; + ComID	Sucht in der Datenbank nach freigegebenen Prüfungen für den Prüfplatz
Prüfplatz einrichten	TLCom.jsp?ComID=[…]&module=SetupTL&testno=[…]	ComID= ABC123 module= SetupTL testno= PNR-P2012113J	Zeilenweise Wertepaare; Unterteilt nach Kategorie beginnend mit <Kategorie> und Ende mit </Kategorie> + ComID	Gibt alle Werte zurück, welche zum Einrichten des Prüfplatzes nötig sind; Soll- + Ist-Werte
Prüfplatz updaten	TLCom.jsp?ComID=[…]&module=UpdateTL&testno=[…]	ComID= ABC123 module= UpdateTL testno= PNR-P2012113J	Erste Zeile: entweder „-No Data Update -“ oder „-Data Update -“ + Zeilenweise Wertepaare; Unterteilt nach Kategorie beginnend mit <Kategorie> und Ende mit </Kategorie> + ComID	Gibt alle Werte zurück, welche zum Updaten des Prüfplatzes nötig sind; nur Soll-Werte
Prüfung aktivieren	TLCom.jsp?ComID=[…]&module=ActivateTest&testno=[…]&activationTimeStamp=[YYYY-mm-dd%20HH:MM:SS]	ComID= ABC123 module= ActivateTest testno= PNR-P2012113J activationTimeStamp=2012-10-17%2015:35:30	ComID	Aktiviert die ausgewählte Prüfung Leerzeichen zwischen Datum und Zeit als “%20” eingeben
Daten Senden Start	TLCom.jsp?ComID=[…]&module=DataStream Start	ComID= ABC123 module= DataStreamStart	SessionID	Bereitet Sammeln der Prüfdaten vor; gibt SessionID, welche im Datenstrom angegeben werden muss
Datenstrom senden	TLCom.jsp;sessionid=[…]?ComID=[…]&module=DataStream&W1=Prüfdatenname1=Prüfdaten&W2=Prüfdatenname2&Prüfdaten	ComID= ABC123 sessionid= 123456789 module= DataStream W1= Zykluszahl%3D7541		Datenübergabe; SessionID muss immer gleich bleiben; W1 – Wn ist beliebig, es werden nur Prüfdatenname und Prüfdaten gespeichert
Daten Senden Ende	TLCom.jsp;sessionid=[…]?ComID=[…]&module=DataStream End	ComID= ABC123 sessionid= 123456789 module= DataStreamEnd	ComID	Daten werden an Datenbank übertragen

Anlage 3, Quellcode TLCom Teil 1

```
<%--
Imports

misc. java classes
--%><%@ page import="java.util.*" %><%--
--%><%@ page import="java.io.*" %><%--
--%><%@ page import="java.text.*" %><%--
--%><%@ page import="java.text.MessageFormat" %><%--

SQLInterface
--%><%@page import="net.miele.de.sqlinterface.*"%><%--
--%><%@page import="net.miele.de.sqlinterface.con.*"%><%--
--%><%@page import="net.miele.de.sqlinterface.dbaccess.*"%><%--
--%><%@page import="net.miele.de.sqlinterface.err.*"%><%--

some ProLab classes
--%><%@ page import="net.miele.de.prolab.connectors.ProLabConnect4QNXSystems" %><%--
--%><%@ page import="net.miele.de.prolab.exceptions.ProLabException" %><%--

Documentation
--%><%
/**
 * <PRE>
 * ##############################################################################
 * ##                                                                          ##
 * ## /System/TL/TLCom.jsp                                                     ##
 * ##                                                                          ##
 * ##                                                                          ##
 * ## The first page of the communication between testlocation and database.##
 * ## It starts with a test for access authorization, using the ComID       ##
 * ## and the Client-Name. After that, it initialize the selected module.   ##
 * ##                                                                          ##
 * ##                                                                          ##
 * ##                                                                          ##
 * ## Request Parameter:                                                       ##
 * ##                                                                          ##
 * ## ComID:    Communication-Id for Identification between testlocation    ##
 * ##           and database                                                   ##
 * ## module:   The name of module      - SearchTest                          ##
 * ##                                   - ActivateTest                        ##
 * ##                                   - SetupTL                             ##
 * ##                                   - UpdateTL                            ##
 * ##                                   - Start                               ##
 * ##                                   - DataStream                          ##
 * ##                                   - End                                 ##
 * ## Depending on the selected module, additional parameters are needed.   ##
 * ## The documentation of the specific includes gives the requested        ##
 * ## parameters.                                                            ##
 * ##                                                                          ##
 * ##                                                                          ##
 * ##############################################################################
 * ## Created                                                                  ##
 * ##  at:     28.09.2012                                                      ##
 * ##  author: Eric Linke                                                      ##
 * ##############################################################################
 * ## Modifications                                                            ##
 * ## 28.09.2012:  Eric Linke                                                  ##
 * ##              - jsp created                                               ##
 * ##############################################################################
 * </PRE>
 *
 * @author Eric Linke
 * @version 0.1.000
 * @since 28.09.2012
 */
%><%--

initialize DataHandlingBean
 --%><jsp:useBean id="DataHandlingBean" class="net.miele.de.prolab.gui.beans.DataHandlingBean" scope="session"></jsp:useBean><%--

database connection
 --%><jsp:useBean     id="DBConPoolObjTL"   class="net.miele.de.sqlinterface.beans.DBConPoolBean" scope="application"><%--
 --%><jsp:setProperty name="DBConPoolObjTL" property="dbConType"
                      value="<%= DBConnectionInformix.IDENT %>" /><%--
 --%><jsp:setProperty name="DBConPoolObjTL" property="dbConHost"
                      value="<%= getServletContext().getInitParameter(\"WEBXML_DB_HOST\") %>" /><%--
 --%><jsp:setProperty name="DBConPoolObjTL" property="dbConServer"
                      value="<%= getServletContext().getInitParameter(\"WEBXML_DB_SERVER\") %>" /><%--
 --%><jsp:setProperty name="DBConPoolObjTL" property="dbConPort"
                      value="<%= getServletContext().getInitParameter(\"WEBXML_DB_PORT\") %>" /><%--
 --%><jsp:setProperty name="DBConPoolObjTL" property="dbConDBName"
                      value="<%= getServletContext().getInitParameter(\"WEBXML_DB_NAME\") %>" /><%--
 --%><jsp:setProperty name="DBConPoolObjTL" property="dbConUser"
                      value="<%= getServletContext().getInitParameter(\"WEBXML_DB_USER\") %>" /><%--
 --%><jsp:setProperty name="DBConPoolObjTL" property="dbConPassword"
                      value="<%= getServletContext().getInitParameter(\"WEBXML_DB_PASSWORD\") %>" /><%--
```

Anlage 4, Quellcode TLCom Teil 2

```
 --%><jsp:setProperty name="DBConPoolObjTL" property="failoverDatabaseConnectionAllowed"
                      value="<%= true %>" /><%--
 --%><jsp:setProperty name="DBConPoolObjTL" property="connectionMaxAge"
                 value="<%= SQLIQuery.toInteger(SQLIQuery.nvl(session.getAttribute(\"ProLabDBConnectionMaxAge\"), 50)) %>" /><%--
 --%></jsp:useBean><%--

final values for page configuration
--%><%      final String SEARCH_TEST        = "SearchTest";
            final String SETUP_TL           = "SetupTL";
            final String UPDATE_TL          = "UpdateTL";
            final String START_SENDING      = "DataStreamStart";
            final String DATASTREAM         = "DataStream";
            final String END_SENDING        = "DataStreamEnd";
            final String ACTIVATE_TEST      = "ActivateTest";
%><%--

 Page Content
--%><%
            int numeration = 0;
            Vector<String> testVector;
            String module = request.getParameter("module");
            String comId = request.getParameter("ComID");
            String testno;
            String time;
            ProLabConnect4QNXSystems proLabConnect = new ProLabConnect4QNXSystems(comId, request.getRemoteHost().toString());
            DBConnection connection = DBConPoolObjTL.getDbCon();
            SQLIQuery comQuery = new SQLIQuery(connection);

            try
            {
                if (proLabConnect.hasAccess(comQuery)) // verify Access
                {
                    //Transform the String-Input in a Integer-Input (1-8) to handle them in an Switch-Case-Procedure
                    if (module.equals(SEARCH_TEST))
                    {
                        %><%@include file="SearchTest.include"%><%
                    }
                    if (module.equals(ACTIVATE_TEST))
                    {
                        %><%@include file="ActivateTest.include"%><%
                    }
                    if (module.equals(SETUP_TL))
                    {
                        %><%@include file="SetupTL.include"  %><%
                    }
                    if (module.equals(UPDATE_TL))
                    {
                        %><%@include file="UpdateTL.include"%><%
                    }
                    if (module.equals(START_SENDING))
                    {
                        %><%@include file="StartSending.include"%><%
                    }
                    if (module.equals(DATASTREAM))
                    {
                        %><%@include file="DataStream.include"%><%
                    }
                    if (module.equals(END_SENDING))
                    {
                        %><%@include file="EndSending.include"%><%
                    }
                    if (module.isEmpty())
                    {
                       throw new Exception("- Missing module-input -");
                    }
                }
                else
                {
                    out.println("- Exception -");
                    out.println("- Access denied -");
                }
            }
            catch (Exception e)
            {
                out.println("- Exception -");
                out.println(e.getMessage());
            }
            finally
            {
                connection.closeConnection();
            }
%>
```

Anlage 5, Quellcode Search Test

```
<%--Documentation
--%><%
/**
 * <PRE>
 * ###############################################################################
 * ##                                                                           ##
 * ## /System/TL/SearchTest.include                                             ##
 * ##                                                                           ##
 * ##                                                                           ##
 * ## This Include searchs for all available Tests und return the               ##
 * ## testnumber and the testtitle.                                             ##
 * ##                                                                           ##
 * ## Request Parameter:                                                        ##
 * ##                                                                           ##
 * ## ComID:    Communication-Id for Identification between testlocation       ##
 * ##           and database                                                    ##
 * ## module:   The name of the module      - SearchTest                        ##
 * ##                                                                           ##
 * ###############################################################################
 * ## Created                                                                   ##
 * ##  at:     28.09.2012                                                       ##
 * ##  author: Eric Linke                                                       ##
 * ###############################################################################
 * ## Modifications                                                             ##
 * ###############################################################################
 * </PRE>
 *
 * @author Eric Linke
 * @version 0.1.000
 * @since 28.09.2012
 */
%><%--

 Page Content
--%><%

    comId=proLabConnect.getAvailableTests(comQuery);
    for(int counter = 1; counter <= comQuery.getNumOfResultRows("AvailableTests"); counter++)
    {
        out.println(comQuery.getResultValue("AvailableTests", "testno", counter) + " "
                                            + comQuery.getResultValue("AvailableTests",
                                            "testtitle", counter));
    }
    out.println("New ComID: " + comId);

%>
```

Anlage 6, Quellcode Activate Test

```
<%--Documentation
--%><%
/**
 * <PRE>
 * ###############################################################################
 * ##                                                                           ##
 * ## /System/TL/ActivateTest.include                                           ##
 * ##                                                                           ##
 * ##                                                                           ##
 * ## This include activate the current test in the database.                   ##
 * ##                                                                           ##
 * ## Request Parameter:                                                        ##
 * ##                                                                           ##
 * ## ComID:                 Communication-Id for identification between       ##
 * ##                        testlocation and database                          ##
 * ## module:                The name of the module      - ActivateTest         ##
 * ## testno:                The testnumber, which should be set up             ##
 * ## activationTimeStamp:   The Activationtime in the following Format:        ##
 * ##                        YY-mm-dd HH:MM:SS                                  ##
 * ##                                                                           ##
 * ###############################################################################
 * ## Created                                                                   ##
 * ##  at:     28.09.2012                                                       ##
 * ##  author: Eric Linke                                                       ##
 * ###############################################################################
 * ## Modifications                                                             ##
 * ###############################################################################
 * </PRE>
 *
 * @author Eric Linke
 * @version 0.1.000
 * @since 28.09.2012
 */
%><%--

Page Content
--%><%
testno = request.getParameter("testno");
time = request.getParameter("activationTimeStamp");
comId=proLabConnect.activateTest(testno, time, DBConPoolObjTL);
out.println("ComID: " + comId);
%>
```

Anlage 7, Quellcode Setup Testlocation

```
<%--Documentation
--%><%
/**
 * <PRE>
 * ###############################################################################
 * ##                                                                           ##
 * ## /System/TL/SetupTL.include                                                ##
 * ##                                                                           ##
 * ##                                                                           ##
 * ## This include deliver all necessary data from the database to the          ##
 * ## testlocation.                                                             ##
 * ##                                                                           ##
 * ## Request Parameter:                                                        ##
 * ##                                                                           ##
 * ## ComID:    Communication-Id for Identification between testlocation       ##
 * ##           and database                                                    ##
 * ## module:   The name of the module      - SetupTL                           ##
 * ## testno:   The testnumber, which should be set up                          ##
 * ##                                                                           ##
 * ###############################################################################
 * ## Created                                                                   ##
 * ##  at:     28.09.2012                                                       ##
 * ##  author: Eric Linke                                                       ##
 * ###############################################################################
 * ## Modifications                                                             ##
 * ###############################################################################
 * </PRE>
 *
 * @author Eric Linke
 * @version 0.1.000
 * @since 28.09.2012
 */
%><%--

Page Content
--%><%
try
{
    testno = request.getParameter("testno");
    comId=proLabConnect.getData4TestSetup(testno, comQuery);

    testVector = proLabConnect.getTestDataInformationPartNames();
    %><%@ include file="PrintData.include" %><%

}
catch (Exception e)
{
    out.println("- Exception -");
    out.println(e.getMessage());
}
finally
{
    out.println("<ComID>");
    out.println(comId);
    out.println("</ComID>\n");
}
%>
```

Anlage 8, Quellcode Update Testlocation Teil 1

```
<%--Documentation
--%><%
/**
 * <PRE>
 * ###############################################################################
 * ##                                                                           ##
 * ## /System/TL/UpdateTL.include                                               ##
 * ##                                                                           ##
 * ##                                                                           ##
 * ## When new data for the testlocation exist, this include deliver them       ##
 * ## from the database to the testlocation. Else it returns a short            ##
 * ## Message                                                                   ##
 * ##                                                                           ##
 * ## Request Parameter:                                                        ##
 * ##                                                                           ##
 * ## ComID:    Communication-Id for Identification between testlocation        ##
 * ##           and database                                                    ##
 * ## module:   The name of the module      - UpdateTL                          ##
 * ## testno:   The testnumber, which should be set up                          ##
 * ##                                                                           ##
 * ###############################################################################
 * ## Created                                                                   ##
 * ##  at:     28.09.2012                                                       ##
 * ##  author: Eric Linke                                                       ##
 * ###############################################################################
 * ## Modifications                                                             ##
 * ###############################################################################
 * </PRE>
 *
 * @author Eric Linke
 * @version 0.1.000
 * @since 28.09.2012
 */
%><%--

Page Content
--%><%

try
{
    testno = request.getParameter("testno");
    comId=proLabConnect.getData4TestUpdate(testno, comQuery);

    testVector = proLabConnect.getTestDataInformationPartNames();

    if (testVector.size() > 0 )
    {
        out.println("- Data Update -");
        %><%@ include file="PrintData.include" %><%
    }
    else
    {
```

Anlage 9, Quellcode Update Testlocation Teil 2

```
53          out.println("- No Data Update -");
54      }
55 }
56 catch (Exception e)
57 {
58      out.println("- Exception -");
59      out.println(e.getMessage());
60 }
61 finally
62 {
63      out.println("<ComID>");
64      out.println(comId);
65      out.println("</ComID>\n");
66 }
67 %>
```

Anlage 10, Quellcode DataStreamStart

```
<%--Documentation
--%><%
/**
 * <PRE>
 * ###############################################################################
 * ##                                                                           ##
 * ## /System/TL/StartSending.include                                           ##
 * ##                                                                           ##
 * ##                                                                           ##
 * ## This include prepares the incomming datastream from the testlocation.    ##
 * ## It returns the session-id.                                                ##
 * ##                                                                           ##
 * ## Request Parameter:                                                        ##
 * ##                                                                           ##
 * ## ComID:    Communication-Id for Identification between testlocation       ##
 * ##           and database                                                    ##
 * ## module:   The name of the module      - Start                             ##
 * ##                                                                           ##
 * ###############################################################################
 * ## Created                                                                   ##
 * ##  at:     28.09.2012                                                       ##
 * ##  author: Eric Linke                                                       ##
 * ###############################################################################
 * ## Modifications                                                             ##
 * ###############################################################################
 * </PRE>
 *
 * @author Eric Linke
 * @version 0.1.000
 * @since 28.09.2012
 */
%><%--

Page Content
--%><%
Hashtable<String, String> hashtable = new Hashtable<String, String>();
session.setAttribute("hashID", hashtable);
out.println("Session-ID = " + session.getId());
%>
```

Anlage 11, Quellcode DataStream Teil 1

```
<%--Documentation
--%><%
/**
 * <PRE>
 * ###############################################################################
 * ##                                                                           ##
 * ## /System/TL/DataStream.include                                             ##
 * ##                                                                           ##
 * ##                                                                           ##
 * ## This include handles the given Data from the testlocation.                ##
 * ## After multiple requests, the data will be stored in one hashtable.        ##
 * ##                                                                           ##
 * ## Request Parameter:                                                        ##
 * ##                                                                           ##
 * ## ComID:    Communication-Id for Identification between testlocation       ##
 * ##           and database                                                    ##
 * ## module:   The name of the module      - DataStream                        ##
 * ##                                                                           ##
 * ###############################################################################
 * ## Created                                                                   ##
 * ##  at:     28.09.2012                                                       ##
 * ##  author: Eric Linke                                                       ##
 * ###############################################################################
 * ## Modifications                                                             ##
 * ###############################################################################
 * </PRE>
 *
 * @author Eric Linke
 * @version 0.1.000
 * @since 28.09.2012
 */
%><%--

Page Content
--%><%
Hashtable<String, String> hash = new Hashtable<String, String>();

Enumeration setEnumeration = request.getParameterNames();
//fill the Enumeration with all Parameternames from the URL

hash = (Hashtable) session.getAttribute("hashID");
//get the Hashtable from the session; necessary for multiple requests
StringBuilder stringbuilder = new StringBuilder();
//the stringbuilder is needed later to put the split-strings together

while (setEnumeration.hasMoreElements())
//do for all Enumeration-Element the following...
{
    String name = (String) setEnumeration.nextElement();
    //initialize for every name and value a "new"-String
    String newname = "";
    String value = request.getParameter(name);
```

Anlage 12, Quellcode DataStream Teil 2

```
53      String newvalue = "";
54
55      if (value.matches(".*=.*"))
56      //only use Parameters, when the value contains an "="-symbol. So the
57      {
58       //module and the Com-Id isn't involved.
59
60          String[] splitArray = value.split("=");
61          //split the value on every "="-Symbol
62          for (int counter = 1; counter < splitArray.length; counter++ )
63          //connect all splitted Strings,excepting the first String
64          //(the new parametername), separated by a "="-Symbol
65          {
66              stringbuilder.append(splitArray[counter]);
67              stringbuilder.append("=");
68              newname = splitArray[0];
69              //the newname is always the first splitArray
70          }
71
72          newvalue = stringbuilder.toString();
73          newvalue = newvalue.substring(0, newvalue.length()-1);
74          //the newvalue is the result of the stringbuilder, cut by
75
76          //the last Symbol (the last "=")
77          hash.put(newname, newvalue);
78          //put the newname and the newvalue into the Hashtable
79          stringbuilder.delete(0, stringbuilder.length());
80          //delete all entrys, between 0 and the own length
81      }
82  }
83  session.setAttribute("hashID", hash);
84  //saves the Hashtable into the session
85  %>
```

Anlage 13, Quellcode DataStreamEnd

```
<%--Documentation
--%><%
/**
 * <PRE>
 * ###############################################################################
 * ##                                                                           ##
 * ## /System/TL/EndSending.include                                             ##
 * ##                                                                           ##
 * ##                                                                           ##
 * ## This sends the complete hashtable to the database.                       ##
 * ##                                                                           ##
 * ## Request Parameter:                                                        ##
 * ##                                                                           ##
 * ## ComID:    Communication-Id for Identification between testlocation       ##
 * ##           and database                                                    ##
 * ## module:   The name of the module      - End                               ##
 * ##                                                                           ##
 * ###############################################################################
 * ## Created                                                                   ##
 * ##  at:     28.09.2012                                                       ##
 * ##  author: Eric Linke                                                       ##
 * ###############################################################################
 * ## Modifications                                                             ##
 * ###############################################################################
 * </PRE>
 *
 * @author Eric Linke
 * @version 0.1.000
 * @since 28.09.2012
 */
%><%--

Page Content
--%><%
Hashtable<String, String> hash2 = new Hashtable<String, String>();
hash2 = (Hashtable) session.getAttribute("hashID");

/**Enumeration getEnumeration = hash2.keys();               //Browser-Output
*while (getEnumeration.hasMoreElements())
*{
*    String strKey = (String) getEnumeration.nextElement();
*    String strValue = (String) hash2.get(strKey);
*    out.print(strKey + "=");
*    out.print(strValue + "\n");
*}
*/
comId=proLabConnect.setTestDataValues(hash2, comQuery);     //Send the Hash

out.println("New ComID: " + comId);
%>
```

Anlage 14, Quellcode Print Data

```
<%--Documentation
--%><%
/**
 * <PRE>
 * ##############################################################################
 * ##                                                                          ##
 * ## /System/TL/PrintData.include                                             ##
 * ##                                                                          ##
 * ##                                                                          ##
 * ## This include provides the output-loop for the SetupTL- and the           ##
 * ## UPdateTL-Include.                                                        ##
 * ##                                                                          ##
 * ##############################################################################
 * ## Created                                                                  ##
 * ##  at:     16.10.2012                                                      ##
 * ##  author: Eric Linke                                                      ##
 * ##############################################################################
 * ## Modifications                                                            ##
 * ##############################################################################
 * </PRE>
 *
 * @author Eric Linke
 * @version 0.1.000
 * @since 16.10.2012
 */
%><%--
 Page Content
--%><%
for(int statementNo = 0; statementNo < testVector.size(); statementNo++)
{
   out.println("<" + testVector.get(statementNo) + ">");
   for(int rowNo = 1; rowNo <= comQuery.getNumOfResultRows(testVector.get
                               (statementNo)); rowNo++)
   {
       for ( int columnNo = 1; columnNo <= comQuery.getNumOfResultColumns
                                          (testVector.get(statementNo)); columnNo++ )
       {
           out.print(comQuery.getResultColumnName(testVector.get(statementNo), columnNo) + "=");
           out.print(SQLIQuery.trim(comQuery.getResultValue(testVector.get
                                            (statementNo), columnNo , rowNo)) + "\n");
       }
   }
   out.println("</" + testVector.get(statementNo) + ">\n");
}

%>
```

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Mittweida, den 24. Januar 2013

Eric Linke